# 68' MICRO JOURNAL

Australia A $4.75 | New Zealand NZ $6.50
Singapore S $9.45 | Hong Kong H $23.50
Malaysia M $9.45 | Sweden 30:-SEK

$2.95 USA

## OS-9 Atari Amiga Mac S-50

6800 6809 68000 68008 68010 68020 68030

This Issue:



OS-9 SK*DOS Atari Amiga FLEX Macintosh

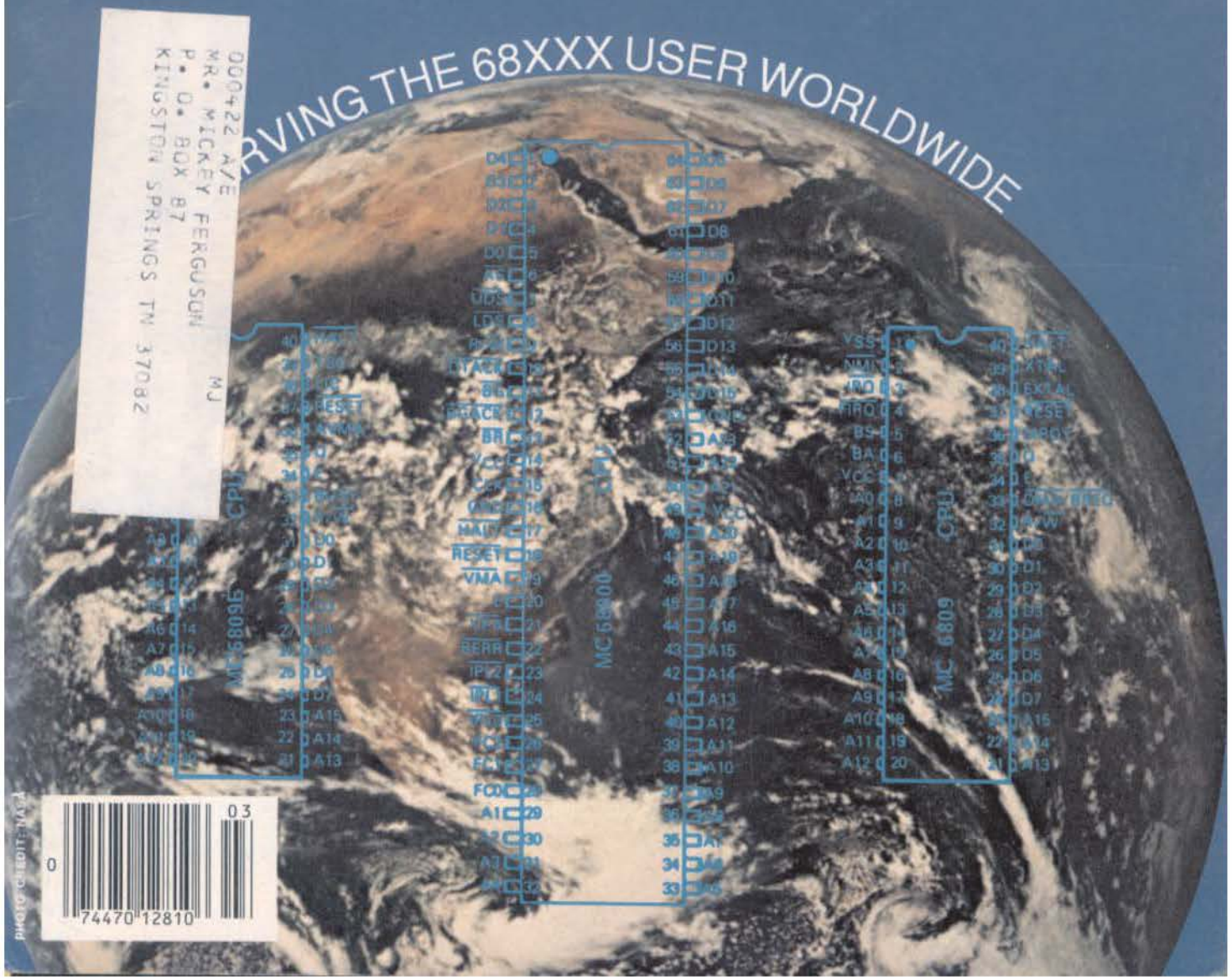
*A User Contributor Journal*

And Lots More!

VOLUME X ISSUE III • Devoted to the 68XXX User • March 1988

The Grandfather of "DeskTop Publishing™"

SERVING THE 68XXX USER WORLDWIDE





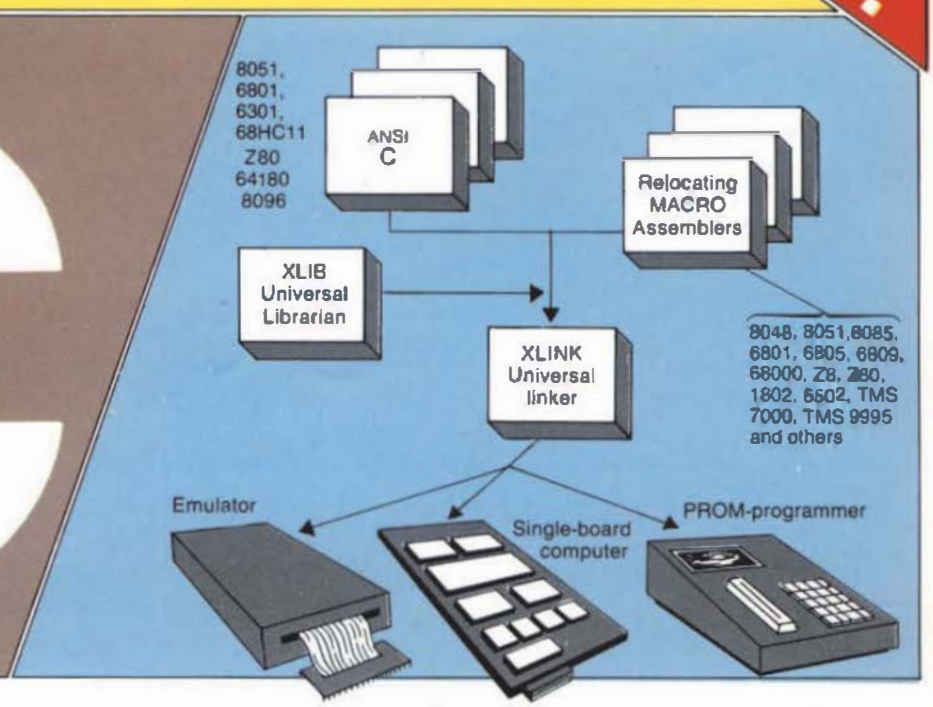

0 74470 12810 03

# ICC C CROSS COMPILER
## FOR OS-9/68xxx BASED SYSTEMS

NEW!

C

8051
6301/6801
68HC11
Z80/64180
8096




### ANSI C

Full implementation of the proposed ANSI standard for C compilers. Includes the Kernighan & Ritchie standard plus improvements for micro-controller development.

### Memory-based compiler

ICC is a fast one-pass compiler based on main memory storage. This has three major advantages:

- NO temporary files
- NO time-consuming assembly pass
- NO separate pre-processor stage

This combines into one single word: *SPEED*

### PROM-able C

ICC makes it possible to put C programs into PROM, still using the full C language and all data types, including type definitions, long integers and statically initialized variables.

### Built-in Type-Checking

ICC has a UNIX LINT-like type-checking option built-in into the compiler. This means that Pascal-like warnings are generated, e.g. when functions are mismatched or undeclared.

### Various Options

- 8051 — single-chip
  - — 64 K CODE + DATA
  - — 64 K CODE + 64 K DATA
- 6301 and 6801
- Z80 and 64180

### Full Package Development System

The ICC compiler package includes:

- C run-time library
- $\mu$-Series Relocatable Macro Assembler
- XLINK Universal Linker
- XLIB Universal Librarian
- Floating-point support
- 150 page manual in three-ring binder

All this together give the micro-controller programmer a powerful Development System Software Environment.

**For more information contact your local distributor:**

Frank Hogg Laboratory
The Regency Tower
Suite 215
770 James Street
Syracuse, NY 13203
Phone: (315)474-7856
Telex: 646740

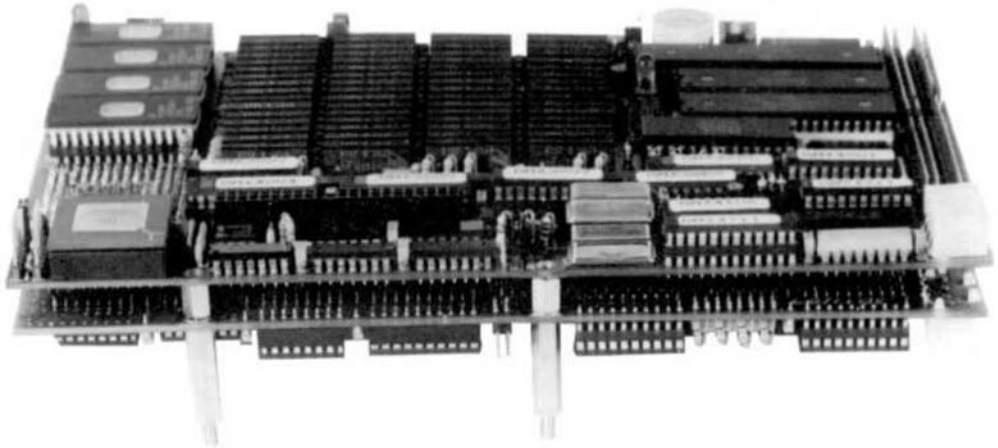
Elsoft AG
Zelweg 12
CH-5405 Baden-Dattwil
Switzerland
Phone: (056) 83 33 77
Telex: 828275

IAR SYSTEMS

OS-9/68xxx version distribution by:

Micromaster Scandinavia AB
Box 1309, S-751 43 UPPSALA, SWEDEN
Tel. int.: + 4618 13 85 95 Telex: 76129

# NOW THE GMX MICRO-20 HAS A TWIN —
# *Announcing The* GMX TWINGLE-20
# 68020 TWIN BOARD COMPUTER WITH MMU



## All the features of the GMX Micro-20, PLUS —

- **2 Megabytes additional RAM — for a total of 4 Megabytes of RAM**
- **8 more Serial ports — for a total of 12, and expandable up to 44.**
- **MEMORY MANAGEMENT UNIT**

**The GMX TWINGLE-20** consists of 2 boards. One of the boards is the same as the Micro-20, except for the 68020 processor which is on the MMU board. It uses the same I/O expansion interface, serial adapter boards, and mounting holes as the GMX Micro-20, making it easy to upgrade existing systems. Any of the currently available GMX Micro-20 I/O expansion boards can be used to provide additional I/O capability. Expansion possibilities include additional serial ports (up to 44 ports total), additional parallel ports, and local area networking of up to 255 GMX Micro-20s and/or TWINGLE-20s.

**The MMU board** contains the additional 2 Megabytes of RAM, 8 serial ports with 2 connectors for the SAB 4 port adaptor cards, and the MMU hardware. The MMU is a proprietary high-speed design that fully supports virtual memory. The system RAM normally operates with only 1 wait-state, regardless of processor speed. An additional wait-state is needed only when program flow crosses a 4K boundary. The MMU can be configured for any one of four different maps, ranging from 8 tasks with 8 megabytes of virtual address space each, to 64 tasks of 1 megabyte each. The MMU can be disabled for applications that do not use hardware memory management.

**The TWINGLE-20** two board set can occupy the same space as a half-height 5.25" disk drive. It is available in 12.5, 16.67 or 20 MHz. versions, and with or without the 68881 FPC.

### SPECIFICATIONS

Size: 8.8 × 5.75 × 1.4 inches.

Power Requirements: +5VDC @ 8.3A typical (20MHz. with 68881).
The TWINGLE-20 itself does not require a +12V supply. +12V supply requirements, if any, are determined by the serial adapter boards and any I/O expansion boards powered through the I/O Expansion Interface.

### SOFTWARE INCLUDED:

An enhanced version of 020Bug with diagnostics for the MMU and the additional RAM and serial ports.

### OPTIONAL SOFTWARE:

**UniFLEX VM,** Virtual Memory version of the UniFLEX operating system which includes all of the features of the GMX Micro-20 version, plus full MMU support.

The UniFLEX VM Operating System is a demand-paged, virtual memory operating system written in 68020 Assembler code for compactness and efficiency. Any UniFLEX system will run faster than a comparable system written in a higher level language. This is important in such areas as context switching, disk I/O, and system call handling. Other features include:

- Compact, efficient Kernel and modules allows handling more users more effectively than UNIX systems, using much less disk space.
- UNIX system V compatibility at the C source code level.
- C Compiler optimized in 68020 code (optional).
- Record locking for shared files.
- Users can share programs in memory.
- Modeled after UNIX systems, with similar commands.
- System accounting facilities.
- Sequential and random file access.
- Maximum record size limited only by the disk size.
- Multiple Level Directories.
- Up to 8 Megabytes of Virtual Memory per user.

All the optional languages and software that run under UniFLEX for the Micro-20 are also available for the TWINGLE-20.

**OS-9** Users can take advantage of the additional RAM and serial ports on the TWINGLE-20. It does not presently support the MMU.



GMX INC 1337 W. 37th Place, Chicago, IL 60609 (312) 927-5510 — TWX 910-221-4055 — FAX (312) 927-7352

A Member of the CPI Family

# 68 Micro Journal

*10 Years of Dedication to Motorola CPU Users*

6800 6809 68000 68010 68020

The Originator of "DeskTop Publishing™"



## Contents



68 MICRO JOURNAL

*"Contribute Nothing - Expect Nothing"* DMW 1986

*COMPUTER PUBLISHING, INC.*

*"Over a Decade of Service"*

"World Wide"

**68 MICRO JOURNAL**
**Computer Publishing Center**
**5900 Cassandra Smith Road**
**PO Box 849**
**Hixson, TN 37343**

**Phone (615) 842-4600 Telex 510 600-6630**



68 Micro Journal is the *original* "DeskTop Publishing" product and has continously published since 1978 using only micro-computers and special "DeskTop" software. Using first a kit built 6800 micro-computer, a modified "ball" typewriter, and "home grown" DeskTop Publishing software. None was commercially available at that time. For over 10 years we have been doing "DeskTop Publishing"! *We originated what has become traditional "DeskTop Publishing"!* Today 68 Micro Journal is acknowledged as the "Grandfather" of "DeskTop Publishing" technology.

---

68 Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage paid ISSN 0194-5025 at Hixson, TN. and additional entries. Postmaster: send form 3597 to 68 Micro Journal, POB 849, Hixson, TN 37343.

### Subscription Rates

1 Year $24.50 USA, Canada & Mexico $34.00 a year. Others add $12.00 a year surface, $48.00 a year Airmail, USA funds. 2 years $42.50, 3 years $64.50 plus additional postage for each additional year.

### Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number, date and a statement that the material is original and the property of the author. Articles submitted should be on diskette, OS-9, SK*DOS, FLEX, Macintosh or MS-DOS. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please! Diagrams o.k.

*Please - do not format with spaces any text indents, charts, etc. (source listing o.k.). We will edit in all formatting. Text should fall flush left and use a carriage return only to indicate a paragraph end. Please write for free authors guide.*

### Letters & Advertising Copy

Letters to the Editor should be the original copy, signed! Letters of gripe as well as praise are acceptable. *We reserve the right to reject any letter or advertising material, for any reason we deem advisable.* Advertising Rates: Commercial please contact 68 Micro Journal Advertising Department. Classified advertising must be non-commercial. Minimum of $15.50 for first 15 words. Add $.60 per word thereafter. No classifieds accepted by telephone.

# OS-9 . . .
# *Making Beautiful Music in the Key of "C"!*

**When** you need to orchestrate beautiful music on your VME system, look to Microware for just the right score. Our finely tuned OS-9 Operating System is truly the maestro's choice when the project requires C Language development. Our superlative C Compiler—also available in an optimized 68020 version—produces fast, compact ROMable code for your most demanding applications. A powerful Assembler, Linker and Symbolic Debugger assists in target development. And our C Compiler is source compatible with UNIX applications and available in cross-compiler configurations for Sun, VAX and Hewlett-Packard environments.

## OS-9 Keeps On Performing Even After the Fat Lady Sings!

Most operating systems hit a sour note when the project reaches completion. But not OS-9. Because of its modular design and UNIX-style architecture, your investment in OS-9 experience, tools and applications translates into a valuable resource for your company's future. OS-9 can be utilized again and again over your entire corporate product range.

## An Accompaniment of Total Support

Microware is proudly setting the industry's standard for customer support. You'll find outstanding technical documentation that leaves nothing in doubt when it comes to real-world applications. A rigorous Quality Assurance program guarantees customer satisfaction by identifying trouble spots before they become customer problems. And with our Customer Hotline, you are only a telephone call away from courteous and concise information. So join the growing legions of Microware "C" aficionados. Call us today and find out how you can create inspiring harmonies on your system.

microware®

MICROWARE SYSTEMS CORPORATION

1900 N.W. 114th Street
Des Moines, IA 50322
Phone 515-224-1929

*Western Regional Office*
4401 Great America Parkway
Santa Clara, CA 95054
Phone 408-980-0201

# MUSTANG -020 Super SBC™


## DATA-COMP Proudly Presents the First Under $4300 "SUPER MICRO" See other advertising (backcover) for economy system (68008) - under $2400 complete!

The MUSTANG-020 68020 SBC provides a powerful, compact, 32 bit computer system featuring the "state of the art" Motorola 68020 "super" micro-processor. It comes standard with 2 megabyte of high-speed SIP dynamic RAM, serial and parallel ports, floppy disk controller, a SASI hard disk interface for intelligent hard disk controllers and a battery backed-up time-of-day clock. Provisions are made for the super powerful Motorola MC68881 floating point math co-processor, for heavy math and number crunching applications. An optional network interface uses one serial (four (4) standard, expandable to 20) as a 125/bit per second network channel. Supports as many as 32 nodes.

The MUSTANG-020 is ideally suited to a wide variety of applications. It provides a cost effective alternative to the other MC68020 systems now available. It is an excellent introductory tool to the world of hi-power, hi-speed new generation "super micros". In practical applications it has numerous applications, ranging from scientific to education. It is already being used by government agencies, labs, universities, business and practically every other critical applications center, worldwide, where true multi-user, multi-tasking needs exist. The MUSTANG-020 is UNIX C level V compatible. Where low cost and power is a must, the MUSTANG-020 is the answer, as many have discovered. Proving that price is not the standard for quality!

As a software development station, a general purpose scientific or small to medium business computer, or a super efficient real-time controller in process control, the MUSTANG-020 is the cost effective choice. With the optional MC68881 floating point math co-processor installed, it has the capability of systems costing many times over it's total acquisition cost.

With the DATA-COMP "total package", consisting of a heavy duty metal cabinet, switching power supply with rf/line by-passing, 5 inch DS/DD 80 track floppy, Xebec hard disk controller, 25 megabyte winchester hard disk, four serial RS-232 ports and a UNIX C level V compatible multi-tasking, multi-user operating system, the price is under $4300, w/12.5 megahertz system clock (limited time offer). Most all popular high level languages are available at very reasonable cost. The system is expandable to 32 serial ports, at a cost of less than $65 per port, in multiples of 8 port expansion options.

The SBC fully populated, quality tested, with 4 serial ports pre-wired and board mounted is available for less that $2500. Quantity discounts are available for OEM and special applications, in quantity. All that is required to bring to complete "system" standards is a cabinet, power supply, disks and operating system. All these are available as separate items from DATA-COMP.

*Available 12.5 - 25 Mhz systems, call for special prices*

A special version of the Motorola 020-BUG is installed on each board. 020-BUG is a ROM based debugger package with facilities for downloading and executing user programs from a host system. It includes commands for display and modification of memory, breakpoint capabilities, a powerful assembler/disassemble and numerous system diagnostics. Various 020-BUG system routines, such as I/O handlers are available for user programs.

Normal system speed is 3-4.5 MIPS, with burst up to 10 MIPS, at 16.6 megahertz. Intelligent I/O available for some operating systems.

Hands-on "actual experience sessions", before you buy, are available from DATA-COMP. Call or write for additional information or pricing.

## Data-Comp Division

CPI — A Decade of Quality Service™

Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road

Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

## Mustang-020 Mustang-08 Benchmarks

| | 32 bit Integer | Register Long |
|---|---|---|
| IBM AT 7300 Xenix Sys 3 | 9.7 | |
| AT&T 7300 UNIX PC 68010 | 7.2 | 4.3 |
| DEC VAX 11/780 UNIX Berkley 4.2 | 3.6 | 3.2 |
| DEC VAX 11/750 " " | 3.1 | 3.2 |
| 68008 OS-9 68K 8 Mhz | 18.0 | 9.0 |
| 68000 OS-9 68K 10 Mhz | 6.5 | 4.0 |
| ***MUSTANG-08 68008 OS-9 68K 10 Mhz*** | ***9.8*** | ***6.7*** |
| ***MUSTANG-020 68020 OS-9 68K 16 Mhz*** | ***2.2*** | ***0.80*** |
| ***MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz*** | ***1.8*** | ***1.22*** |

```
Main()
        {
            register long i;
            for (i=0; i < 999999; ++i);
        }
```

Estimated MIPS - MUSTANG-020 — 4.5 MIPS,
Burst to 8 - 10 MIPS: Motorola Specs

## Mustang Specifications

12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path
32-bit wide data and address buses, non-multiplexed
on chip instruction cache
object code compatible with all 68XXX family processors
enhanced instruction set - math co-processor interface
68881 math hi-speed floating point co-processor (optional)
direct extension of full 68020 instruction set
full support IEEE P754, draft 10.0
transcendental and other scientific math functions
2 Megabyte of SIP RAM (512 x 32 bit organization)
up to 256K bytes of EPROM (64 x 32 bits)
4 Asynchronous serial I/O ports standard
optional to 20 serial ports
standard RS-232 interface
optional network interface
Xtra 8 bit parallel port (1/2 MC68230)
Centronics type pinout
expansion connector for I/O devices
16 bit data path
256 byte address space
2 interrupt inputs
clock and control signals
Motorola I/O Channel Modules
time of day clock/calendar w/battery backup
controller for 2, 5 1/4" floppy disk drives
single or double side, single or double density
35 to 80 track selectable (48.96 TPI)
SASI interface
programmable periodic interrupt generator
interrupt rate from micro-seconds to seconds
highly accurate time base (5 PPM)
5 bit sense switch, readable by the CPU
Hardware single-step capability


***Don't be mislead! ONLY Data-Comp delivers the Super MUSTANG-020***

### OS-9

| | |
|---|---|
| OS-9 Professional Ver *Includes C Compiler | $850.00 |
| Basic09 | 300.00 |
| C Compiler | 500.00 |
| 68000 Disassembler (w/source add: $100.00) | 100.00 |
| Fortran 77 | 750.00 |
| Microware Pascal | 500.00 |
| Oregonsoft Pascal | 900.00 |
| Stylo-Graph | 495.00 |
| Stylo-Spell | 195.00 |
| Stylo-Merge | 175.00 |
| Stylo-Graph-Spell-Merge | 695.00 |
| PAT w/C source | 229.00 |
| fUST w/C source | 79.95 |
| PAT/fUST Combo | 249.50 |
| Sculptor+ (see below) | 995.00 |
| COM | 125.00 |

### UniFLEX

| | |
|---|---|
| UniFLEX (68020 ver) | $450.00 |
| Screen Editor | 150.00 |
| Sort-Merge | 200.00 |
| BASIC/PreCompiler | XXX.00 |
| C Compiler | 350.00 |
| COBOL | 750.00 |
| CMODEM w/source | 100.00 |
| TMODEM w/source | 100.00 |
| X-TALK (see Ad) | 99.95 |
| Cross Assembler | 50.00 |
| Fortran 77 | 450.00 |
| Sculptor+ (see below) | 995.00 |
| Standard MUSTANG-020™ shipped 12.5 Mhz. | |
| Add for 16.6 Mhz 68020 | 375.00 |
| Add for 16.6 Mhz 68881 | 375.00 |
| Add for 20 Mhz 68020/RAM | 750.00 |
| 16 Port exp. RS-232 | 335.00 |
| Requires 1 or 2 Adapter Cards below RS232 Adapter | 165.00 |
| Each card supports 4 additional ser. ports (total of 36 serial ports supported) | |
| 60 line Parallel I/O card Uses 3 68230 Interface/Timer chips, 6 groups of 8 lines each, separate buffer direction control for each group. | 398.00 |
| Prototype Board area for both dip and PGA devices & a pre-wired memory area up to 512K DRAM. | 75.00 |
| SBC-AN Interface between the system and ARCNET certified token passing LAN, fiber optics optional - call. LAN software drivers | 475.00 120.00 |
| Expansion for Motorola I/O Channel Modules Special for complete MUSTANG-020™ system buyers - Sculptor+ $695.00. SAVE $300.00 | $195.00 |
| Software Discounts | |

All MUSTANG-020™ system and board buyers are entitled to discounts on all listed software: 10-70% depending on item. Call or write for quotes. Discounts apply after the sale as well.

The **PRO!**

These hi-speed 68020 systems are presently working at NASA, Atomic Energy Commission, Government Agencies as well as Universities, Business, Labs, and other Critical Applications Centers, worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level V compatibility and low cost is a must.

Only the "PRO" Version of OS-9 Supported!

This is *HEAVY DUTY* Country!

For a limited time we will offer a $400 trade-in on your old MUSTANG SBC. Must be working properly and complete with all software, cables and documentation. Call for more information

**Price List:**

| | | |
|---|---|---|
| Mustang-020 SBC | | $2490.00 |
| Cabinet w/switching PS | | $299.95 |
| 5"-80 track floppy DS/DD | | $269.95 |
| Floppy Cable | | $39.95 |
| OS-9 68K Professional Version | | $850.00 |
| C Compiler ($500 Value) | | N/C |
| Winchester Cable | | $39.95 |
| Winchester Drive 25 Mbyte | | $895.00 |
| Hard Disk Controller | | $395.00 |
| Shipping USA UPS | | $20.00 |
| UniFLEX | Less | $100.00 |
| MC68881 f/p math processor | Add | $275.00 |
| 16.67 Mhz MC68020 | | $375.00 |
| 16.67 Mhz MC68881 | | $375.00 |
| 20 Mhz MC68020 Sys | | $750.00 |
| Note all 68881 chips work with 20 Mhz Sys | Total: | $5299.80 |

# Save $1000.00

# Complete 25 Mbyte HD System $4299.80

# 85Mbyte HD System $5748.80

Note: Only Professional OS-9 Now Available (68020 Version)
Includes ($500) C Compiler - 68020 & 68881 Supported -
For UPGRADES Write or Call for Professional OS-9 Upgrade Kit

**Data-Comp Division**

A Decade of Quality Service
Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

# PAT - JUST

**PAT**

**With 'C' Source**

**$229.00**

All OS-9 68XXX Systems


**PAT FROM S. E. MEDIA** -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved **PIE**, this is for **YOU! All PIE features & much more!** Too many features to list. And if you don't like ours, change or add your own. **C source included.** Easily configured to your CRT terminal, with special configuration section. No sweat!

**68008 - 68000 - 68010 - 68020 OS-9 68K $229.00**

# COMBO PAT/JUST

# Special $249.00

## JUST

**JUST from S. E. MEDIA - -** Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

**68008 - 68000 - 68010 - 68020 OS-9 68K**
**With 'C' source $79.95**

*The C Programmers Reference Source. Always Right On Target!*

# C User Notes

**A Tutorial Series**

By: Dr. E. M. 'Bud' Pass
1454 Latta Lane N.W.
Conyers, GA 30207
404 483-1717/4570
*Computer Systems Consultants*

## INTRODUCTION

This chapter concludes the discussion and presentation of a public-domain portable math library written in C by Fred Fish.

## MATH LIBRARY

**The log.c function returns the natural logarithm of its argument.**

```
/*
 *      log   double precision natural log
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

static double    log_pcoeffs[] =
{
    -0.240139179559210509B6e2,
    0.309572928215376500b2e2,
    -0.963769093368686593246e1,
    0.4210873712179797145
};

static double    log_qcoeffs[] =
{
    -0.120069589779605254 71e2,
    0.194809660700889730 51e2,
    -0.891110902793783123 37e1,
    1.0000
};

static char    funcname[] = "log";

double    log (x)
double    x;
{
    auto int k;
    auto double s;
    auto double z;
    auto double zt2;
    auto double pqofz;
    auto struct exception xcpt;
    extern double    frexp ();
    extern double    poly ();

    DBUG_ENTER (funcname);
    DBUG_3 ("login", "arg %le", x);
    if (!x)
    {
        xcpt.type = SING;
        xcpt.name = funcname;
        xcpt.arg1 = x;
        if (!matherr (&xcpt))
        {
            fprintf (stderr, "%s: SINGULARITY error\n",
funcname);
            errno = EDOM;
            xcpt.retval = -MAXDOUBLE;
        }
    }
    else
    if (x < 0.0)
    {
        xcpt.type = DOMAIN;
        xcpt.name = funcname;
        xcpt.arg1 = x;
        if (!matherr (&xcpt))
        {
            fprintf (stderr, "%s: DOMAIN error\n",
funcname);
            errno = EDOM;
            xcpt.retval = -MAXDOUBLE;
        }
    }
    else
    {
        s = SQRT2 * frexp (x, &k);
        DBUG_3 ("log", "k = %d", k);
        DBUG_3 ("log", "s = %le", s);
        z = (s - 1.0) / (s + 1.0);
        DBUG_3 ("log", "z = %le", z);
        zt2 = z * z;
        DBUG_3 ("log", "zt2 = %le", zt2);
        pqofz = z * (poly (3, log_pcoeffs, zt2) /
            poly (3, log_qcoeffs, zt2));
        DBUG_3 ("pqofz", "pqofz = %le", pqofz);
        DBUG_3 ("log", "k = %d", k);
        DBUG_3 ("log", "LN2 = %le", LN2);
        x = k * LN2;
        DBUG_3 ("log", "x = %le", x);
        x -= INSQRT2;
        DBUG_3 ("log", "x = %le", x);
        x += pqofz;
        DBUG_3 ("log", "x = %le", x);
        xcpt.retval = x;
    }
    DBUG_3 ("logout", "result %le", xcpt.retval);
    DBUG_RETURN (xcpt.retval);
}
```

**The log10.c function returns the common logarithm of its argument.**

```
/*
 *      log10   double precision common log
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

static char     funcname[] = "log10";

double     log10 (x)
double     x;
{
    extern double     log ();

    DBUG_ENTER (funcname);
    DBUG_3 ("log10in", "arg %le", x);
    x = LOG10E * log (x);
    DBUG_3 ("log10out", "result %le", x);
    DBUG_RETURN (x);
}
```

**The max.c function returns the larger of its two arguments.**

```
/*
 *      max   double precision maximum of two arguments
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

double     max (x, y)
double     x;
double     y;
{
    ENTER ("max");
    DEBUG4 ("maxin", "x = %le  y = %le", x, y);
    if (x < y)
    {
        x = y;
    }
    DEBUG3 ("maxout", "result %le", x);
    LEAVE ();
    return (x);
}
```

**The min.c function returns the smaller of its two arguments.**

```
/*
 *      min   double precision minimum of two arguments
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

double     min (x, y)
double     x;
double     y;
{
    ENTER ("min");
    DEBUG4 ("minin", "x = %le  y = %le", x, y);
    if (x > y)
    {
        x = y;
    }
    DEBUG3 ("minout", "result %le", x);
    LEAVE ();
    return (x);
}
```

**The mod.c function returns the remainder after division of its first argument by its second argument.**

```
/*
 *      mod   double precision modulo
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

double     mod (value, base)
double     value;
double     base;
{
    auto double intpart;
    extern double     modf ();

    DBUG_ENTER ("mod");
    DBUG_4 ("modin", "args %le %le", value, base);
    if (base)
        value /= base;
    value = modf (value, &intpart);
    value *= base;
    DBUG_3 ("modout", "result %le", value);
    DBUG_RETURN (value);
}
```

**The poly.c function evaluates a polynomial and its value. Its three arguments are the order of the polynomial, a pointer to an array of double precision polynomial coefficients (in ascending order), and the independent variable.**

```
/*
 *      poly   double precision polynomial evaluation
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

double     poly (order, coeffs, x)
register int     order;
double     *coeffs;
double     x;
{
    auto double curr_coeff;
    auto double rtn_value;

    DBUG_ENTER ("poly");
    DBUG_5 ("polyin", "args %d %#x %le", order, coeffs,
x);
    if (order <= 0)
    {
        rtn_value = *coeffs;
```

```
    }
    else
    {
        curr_coeff = *coeffs;    /* Bug in Unisoft's
compiler. */
        coeffs++;            /* Bad code gen for *coeffs++.
*/
        rtn_value = curr_coeff + x * poly (--order,
coeffs, x);
    }
    DBUG_3 ("polyout", "result %le", rtn_value);
    DBUG_RETURN (rtn_value);
}
```

**The scale.c function adds a specified integer to a number's exponent, effectively multiplying by a power of 2 for positive scale values and dividing by a power of 2 for negative scale values.**

```
/*
 *      scale   scale a double precision number by power
of 2
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

#ifdef pdp11
#define EXP_MASK 0x7F800000  /* Mask for exponent       */

#define MANT_MASK 0x807FFFFF /* Mask for mantissa       */

#define EXP_SHIFTS 23        /* Shifts to get into LSB's
*/
#define LEXP_MASK 0377       /* Mask for shifted exponent
*/
#endif

#ifdef mc68000
#define EXP_MASK 0x7F800000  /* Mask for exponent       */

#define MANT_MASK 0x807FFFFF /* Mask for mantissa       */

#define EXP_SHIFTS 23        /* Shifts to get into LSB's
*/
#define LEXP_MASK 0377       /* Mask for shifted exponent
*/
#endif

static union
{
    double    dval;
    long      lval[2];
}   share;

double    scale(value, scale)
double    value;
register int    scale;
{
    register long    temp1, temp2, *lpntr;

    lpntr = &share.lval[0];
    share.dval = value;
    temp1 = *lpntr;
    temp2 = ((temp1 & EXP_MASK) >> EXP_SHIFTS) + scale;
    if (temp2 > MAX_EXPONENT + 128)
    {
        pmlerr(SCALE_OVERFLOW);
    }
    else
    if (temp2 < 0)
    {
        pmlerr(SCALE_UNDERFLOW);
    }
    else
    {
        temp1 &= MANT_MASK;
        temp2 = temp2 << EXP_SHIFTS;
        *lpntr = temp1 | temp2;
    }
    return (share.dval);
}
```

**The mod.c function returns its first argument with the same sign as its second argument.**

```
/*
 *      sign    transfer of sign
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

double    sign (x, y)
double    x;
double    y;
{
    double    rtnval;

    ENTER ("sign");
    DEBUG4 ("signin", "args %le %le", x, y);
    if (x > 0.0)
    {
        if (y > 0.0)
        {
            rtnval = x;
        }
        else
        {
            rtnval = -x;
        }
    }
    else
    {
        if (y < 0.0)
        {
            rtnval = x;
        }
        else
        {
            rtnval = -x;
        }
    }
    DEBUG3 ("signout", "result %le", rtnval);
    LEAVE ();
    return (rtnval);
}
```

**The sin.c function returns the sine of its argument.**

```
/*
 *      sin     double precision sine
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

static double    sin_pcoeffs[] =
{
    0.20664343336995858240e7,
    -0.18160398797407332550e6,
    0.35999306949636188317e4,
    -0.20107483294588615719e2
};

static double    sin_qcoeffs[] =
{
    0.26310659102647698963e7,
    0.39270242774649000308e5,
    0.27811919481083844087e3,
    1.0
};

static char    funcname[] = "sin";

double    sin (x)
double    x;
{
    double    y;
    double    yt2;
    double    rtnval;
    extern double    mod ();
    extern double    cos ();
    extern double    poly ();
    auto struct exception xcpt;

    DBUG_ENTER (funcname);
    DBUG_3 ("sinin", "arg %le", x);
    if (x < -PI || x > PI)
    {
        x = mod (x, TWOPI);
        if (x > PI)
        {
            x = x - TWOPI;
        }
        else
        if (x < -PI)
        {
            x = x + TWOPI;
        }
    }
    if (x > HALFPI)
    {
        xcpt.retval = -(sin (x - PI));
    }
    else
    if (x < -HALFPI)
    {
        xcpt.retval = -(sin (x + PI));
    }
    else
    if (x > FOURTHPI)
    {
        xcpt.retval = cos (HALFPI - x);
    }
    else
    if (x < -FOURTHPI)
    {
        xcpt.retval = -(cos (HALFPI + x));
    }
    else
    if (x < X6_UNDERFLOWS && x > -X6_UNDERFLOWS)
    {
        xcpt.retval = x;
    }
    else
    {
        y = x / FOURTHPI;
        yt2 = y * y;
        xcpt.retval = y * (poly (3, sin_pcoeffs, yt2) /
            poly(3, sin_qcoeffs, yt2));
    }
    DBUG_3 ("sinout", "result %le", xcpt.retval);
    DBUG_RETURN (xcpt.retval);
}
```

**The sinh.c function returns the hyperbolic sine of its argument.**

```
/*
 *      sinh    double precision hyperbolic sine
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

static char    funcname[] = "sinh";

double    sinh (x)
double    x;
{
    extern double    exp ();
    auto struct exception xcpt;

    DBUG_ENTER (funcname);
    DBUG_3 ("sinhin", "arg %le", x);
    if (x > LOGE_MAXDOUBLE)
    {
        xcpt.type = OVERFLOW;
        xcpt.name = funcname;
        xcpt.arg1 = x;
        if (!matherr (&xcpt))
        {
            fprintf (stderr, "%s: OVERFLOW error\n",
funcname);
            errno = ERANGE;
            xcpt.retval = MAXDOUBLE;
        }
    }
    else
    if (x < LOGE_MINDOUBLE)
    {
        xcpt.type = UNDERFLOW;
        xcpt.name = funcname;
        xcpt.arg1 = x;
        if (!matherr (&xcpt))
        {
            fprintf (stderr, "%s: UNDERFLOW error\n",
funcname);
            errno = ERANGE;
            xcpt.retval = MINDOUBLE;
        }
    }
    else
    {
```

```
        x = exp (x);
        xcpt.retval = 0.5 * (x - (1.0 / x));
    }
    DBUG_3 ("sinhout", "result %le", xcpt.retval);
    DBUG_RETURN (xcpt.retval);
}
```

**The sqrt.c function returns the square root of its argument.**

```
/*
 *      sqrt    double precision square root
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

#define P0 0.594604482  /* Approximation coeff  */
#define P1 2.54164041   /* Approximation coeff  */
#define Q0 2.13725758   /* Approximation coeff  */
#define Q1 1.0          /* Approximation coeff  */

#define ITERATIONS 3    /* Number of iterations */

static char     funcname[] = "sqrt";

double    sqrt (x)
double    x;
{
    auto int k;
    register int    bugfix;
    register int    kmod2;
    register int    count;
    auto int exponent;
    auto double m;
    auto double u;
    auto double y;
    auto double rtnval;
    auto struct exception xcpt;
    extern double    frexp ();
    extern double    ldexp ();

    DBUG_ENTER ("sqrt");
    DBUG_3 ("sqrtin", "arg %le", x);
    if (!x)
    {
        rtnval = 0.0;
    }
    else
    if (x < 0.0)
    {
        xcpt.type = DOMAIN;
        xcpt.name = funcname;
        xcpt.arg1 = x;
        if (!matherr (&xcpt))
        {
            fprintf (stderr, "%s: DOMAIN error\n",
funcname);
            errno = EDOM;
            xcpt.retval = 0.0;
        }
    }
    else
    {
        m = frexp (x, &k);
        u = (P0 + (P1 * m)) / (Q0 + (Q1 * m));
        for (count = 0, y = u; count < ITERATIONS;
count++)
        {
            y = 0.5 * (y + (m / y));
        }
        if ((kmod2 = (k % 2)) < 0)
        {
            y /= SQRT2;
        }
        else
        if (kmod2 > 0)
        {
            y *= SQRT2;
        }
        bugfix = 2;
        xcpt.retval = ldexp (y, k / bugfix);
    }
    DBUG_3 ("sqrtout", "result %le", xcpt.retval);
    DBUG_RETURN (xcpt.retval);
}
```

**The tan.c function returns the tangent of its argument.**

```
/*
 *      tan    Double precision tangent
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

static char     funcname[] = "tan";

double    tan (x)
double    x;
{
    double    sinx;
    double    cosx;
    auto struct exception xcpt;
    extern double    sin ();
    extern double    cos();

    DBUG_ENTER (funcname);
    DBUG_3 ("tanin", "arg %le", x);
    sinx = sin (x);
    cosx = cos (x);
    if (!cosx)
    {
        xcpt.type = OVERFLOW;
        xcpt.name = funcname;
        xcpt.arg1 = x;
        if (!matherr (&xcpt))
        {
            fprintf (stderr, "%s: OVERFLOW error\n",
funcname);
            errno = ERANGE;
            if (sinx >= 0.0)
            {
                xcpt.retval = MAXDOUBLE;
            }
            else
            {
                xcpt.retval = -MAXDOUBLE;
            }
        }
    }
    else
    {
        xcpt.retval = sinx / cosx;
    }
    DBUG_3 ("tanout", "result %le", xcpt.retval);
    DBUG_RETURN (xcpt.retval);
}
```

**The tanh.c function returns the hyperbolic tangent of its argument.**

```
/*
 *      tanh    double precision hyperbolic tangent
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

static char     funcname[] = "tanh";

double    tanh (x)
double    x;
{
    auto struct exception xcpt;
    register int     positive;
    extern double     sinh ();
    extern double     cosh ();

    DBUG_ENTER (funcname);
    DBUG_3 ("tanhin", "arg %le", x);
    if (x > TANH_MAXARG || x < -TANH_MAXARG)
    {
        if (x > 0.0)
        {
            positive = 1;
        }
        else
        {
            positive = 0;
        }
        xcpt.type = PLOSS;
        xcpt.name = funcname;
        xcpt.arg1 = x;
        if (!matherr (&xcpt))
        {
                fprintf (stderr, "%s: PLOSS error\n",
funcname);
            errno = ERANGE;
            if (positive)
            {
                xcpt.retval = 1.0;
            }
            else
            {
                xcpt.retval = -1.0;
            }
        }
    }
    else
    {
        xcpt.retval = sinh (x) / cosh (x);
    }
    DBUG_3 ("tanhout", "result %le", xcpt.retval);
    return (xcpt.retval);
}
```

**The xexp.c function returns the exponent of its argument.**

```
/*
 *      xexp   extract double precision number's exponent
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

#ifdef mc68000
#ifdef IEEE
#define EXP_MASK 0x7FF00000  /* Mask for exponent      */

#define EXP_SHIFTS 20        /* Shifts to get into LSB's
*/
#define EXP_BIAS 1023       /* Exponent bias          */

#else
#define EXP_MASK 0x7F800000  /* Mask for exponent      */

#define EXP_SHIFTS 23        /* Shifts to get into LSB's
*/
#define EXP_BIAS 128        /* Exponent bias          */

#endif
#endif

#ifdef pdp11
#define EXP_MASK 0x7F800000  /* Mask for exponent      */

#define EXP_SHIFTS 23        /* Shifts to get into LSB's
*/
#define EXP_BIAS 128        /* Exponent bias          */

#endif

union dtol
{
    double    dval;
    int     ival[2];
};

int     xexp (value)
union dtol value;
{
    register int     *ipntr;

    if (!value.dval)
    {
        return (0);
    }
    ipntr = &value.ival[0];
    *ipntr &= EXP_MASK;
    *ipntr >>= EXP_SHIFTS;
    *ipntr -= EXP_BIAS;
    return (*ipntr);
}
```

**The xexp.c function returns the mantissa of its argument.**

```
/*
 *    xmant  extract double precision number's mantissa
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

#ifdef pdp11
#define MANT_MASK 0x807FFFFF /* Mantissa extraction
mask    */
#define ZPOS_MASK 0x40000000 /* Positive # mask for exp
= 0 */
#define ZNEG_MASK 0x40000000 /* Negative # mask for exp
= 0 */
#endif

#ifdef mc68000
#ifdef IEEE
#define MANT_MASK 0x800FFFFF /* Mantissa extraction
mask    */
#define ZPOS_MASK 0x3FF00000 /* Positive # mask for exp
= 0 */
#define ZNEG_MASK 0x3FF00000 /* Negative # mask for exp
= 0 */
#else
#define MANT_MASK 0x807FFFFF /* Mantissa extraction
mask    */
#define ZPOS_MASK 0x40000000 /* Positive # mask for exp
= 0 */
#define ZNEG_MASK 0x40000000 /* Negative # mask for exp
= 0 */
#endif
#endif

union dtol
{
    double    dval;
    int    ival[2];
};

double    xmant (value)
union dtol value;
{
    register int    *ipntr;

    ipntr = &value.ival[0];
    *ipntr &= MANT_MASK;
    *ipntr |= ZPOS_MASK;
    return (value.dval);
}
```

## EXAMPLE C PROGRAM

Following is this month's example C program; it tests the functions in the math library which accept one complex argument and return a complex result. Other cases would be tested in a similar manner.

```
/*
 *      c2c.c   test complex to complex math functions
 */

#include <stdio.h>
#include "pmluser.h"
#include <dbug.h>

#define TRUE 1
#define FALSE 0
#define MAX_ABS_ERR 1.0e-6 /* catch only bad errors */


static int    vflag; /* Flag for verbose option */
static int   eflag; /* Simulate an error to error printout
*/
static int    sflag; /* Flag to show final statistics */


static double    max_abs_err = MAX_ABS_ERR;

/*
 *      External functions which are used internally.
 */

extern char    *strtok ();
extern double    atof ();
extern double    cabs ();
extern COMPLEX csubt ();
extern COMPLEX cdiv ();

/*
 *      External functions to be tested.
 */

extern COMPLEX cacos();
extern COMPLEX casin();
extern COMPLEX catan();
extern COMPLEX ccos();
extern COMPLEX ccosh();
extern COMPLEX cexp();
extern COMPLEX clog();
extern COMPLEX crcp();
extern COMPLEX csin();
extern COMPLEX csinh();
extern COMPLEX csqrt();
extern COMPLEX ctan();
extern COMPLEX ctanh();

/*
 *    Define all recognized test functions.  Each function
 *       must have an entry in this table, where each
 *       entry contains the information specified in the
 *       structure "test".
 *
 */

struct test
{           /* Structure of each function to be tested
*/
    char    *name;      /* Name of the function to test
*/
```

```
    COMPLEX (*func)(); /* Pointer to the function's
entry point */
    double    max_err; /* Error accumulator for this
function */
);

static struct test tests[] =
{  /* Table of all recognized functions */
    "cacos", cacos, 0.0,        /* Complex arc cosine
*/
    "casin", casin, 0.0,        /* Complex arc sine */

    "catan", catan, 0.0,        /* Complex arc tangent
*/
    "ccos", ccos, 0.0,          /* Complex cosine */
    "ccosh", ccosh, 0.0,        /* Complex hyperbolic
cosine */
    "cexp", cexp, 0.0,          /* Complex exponential
*/
   "clog", clog, 0.0,       /* Complex natural logarithm
*/
    "crcp", crcp, 0.0,          /* Complex reciprocal
*/
    "csin", csin, 0.0,          /* Complex sine */
    "csinh", csinh, 0.0,        /* Complex hyperbolic
sine */
    "csqrt", csqrt, 0.0,        /* Complex square root
*/
    "ctan", ctan, 0.0,          /* Complex tangent */

    "ctanh", ctanh, 0.0,        /* Complex hyperbolic
tangent */
   NULL, NULL, 0.0          /* Function list end marker
*/
);

/*
 *      main   entry point for c2c test utility
 */

main (argc, argv)
int    argc;
char    *argv[];
{
    ENTER ("main");
    DEBUGWHO (argv[0]);
    options (argc, argv);
    dotests (argv);
    statistics ();
    LEAVE ();
}

/*
 *    dotests   process each test from stdin directives
 */

dotests (argv)
char    *argv[];
{
    char    buffer[256];        /* Directive buffer */

    char    function[64];       /* Specified function
name */
    COMPLEX argument;           /* Specified function
argument */
    COMPLEX expected;           /* Specified expected
result */
    COMPLEX result;             /* Actual result */
   COMPLEX error;             /* Relative or absolute
error */
   double   abs_err;        /* Absolute value of error
*/
    struct test *testp;         /* Pointer to function
test */
   struct test *lookup ();     /* Returns function test
pointer */
    register char    *strp;     /* Pointer to next token
in string */

    ENTER ("dotests");
    while (fgets (buffer, sizeof(buffer), stdin))
    {
        strcpy (function, "(null)");
        argument.real = argument.imag = 0.0;
        expected.real = expected.imag = 0.0;
        sscanf (buffer, "%s %le %le %le %le",
            function, &argument.real, &argument.imag,
            &expected.real, &expected.imag);
        testp = lookup (function);
        if (testp == NULL)
        {
            fprintf (stderr, "%s: unknown function
\"%s\".\n",
            argv[0], function);
        }
        else
        {
            result = (*testp ->func)(argument);
            if (vflag)
            {
               printf ("%s(%le + j %le) \n   = %30.23le
+ j %30.23le.\n",
                                function, argument.real,
argument.imag, result.real,
                result.imag);
            }
            if (expected.real || expected.imag)
            {
                error = csubt (result, expected);
                error = cdiv (error, expected);
            }
            else
            {
                error = result;
            }
            abs_err = cabs (error);
            if ((abs_err > max_abs_err) || eflag)
            {
                fprintf (stderr,
                    "%s: error in \"%s\"\n", argv[0],
function);
                fprintf (stderr,
                        "\treal (arg)\t\t%25.20le\n",
argument.real);
                fprintf (stderr,
                        "\timag (arg)\t\t%25.20le\n",
argument.imag);
                fprintf (stderr,
                      "\treal (result)\t\t%25.20le\n",
result.real);
                fprintf (stderr,
                      "\timag (result)\t\t%25.20le\n",
result.imag);
                fprintf (stderr,
                    "\treal (expected)\t\t%25.20le\n",
expected.real);
                fprintf (stderr,
                    "\timag (expected)\t\t%25.20le\n",
expected.imag);
            }
            if (abs_err > testp ->max_err)
            {
                testp ->max_err = abs_err;
            }
        }
```

```
    }
    LEAVE ();
}

/*
 *      options   process command line options
 */

options (argc, argv)
int    argc;
char    *argv[];
{
    register int    flag;
    extern int    getopt ();
    extern char    *optarg;

    ENTER ("options");
    eflag = sflag = vflag = FALSE;
    while ((flag = getopt (argc, argv, "#:el:sv")) !=
EOF)
    {
        switch (flag)
        {
        case '#':
            DEBUGPUSH (optarg);
            break;
        case 'e':
            eflag = TRUE;
            break;
        case 'l':
            sscanf (optarg, "%le", &max_abs_err);
               DEBUG3 ("args", "max_abs_err = %le",
max_abs_err);
            break;
        case 's':
            sflag = TRUE;
            break;
        case 'v':
            vflag = TRUE;
            break;
        }
    }
    LEAVE ();
}

/*
 *      loopup   lookup test in known test list
 */

struct test *lookup (funcname)
char    *funcname;
{
    struct test *testp;
    struct test *rtnval;

    ENTER ("lookup");
    rtnval = (struct test *) NULL;
   for (testp = tests; testp ->name && !rtnval; testp++)
    {
        if (!strcmp (testp ->name, funcname))
        {
            rtnval = testp;
        }
    }
    LEAVE ();
    return (rtnval);
}

/*
 *      statistics   print final statistics if desired
 */

statistics ()
{
    struct test *tp;

    ENTER ("statistics");
    if (sflag)
    {
        for (tp = tests; tp ->name; tp++)
        {
                printf ("%s:\tmaximum relative error
%le\n",
            tp ->name, tp ->max_err);
        }
    }
    LEAVE ();
}
```

**EOF**

# Basically OS-9

*A Tutorial Series*

By: Ron Voigts
2024 Baldwin Court
Glendale Heights, IL

## SOLVING YOUR DEBUG PROBLEMS

Last month I dealt with the "bugs" in programs. Debugging a program is probably one of the most important aspects of being a programmer. As many of you know I am an engineer by profession. So besides dealing with software, I also deal with hardware. I have worked on many hardware problems. Even the best designs do not always work the first time out. (Or the second time , or the third time, ... ) Trouble shooting a circuit involves walking through it step-by-step. It involves injecting signals to try a particular portion of the circuit. And many times, it means using some pretty sophisticated instrumentation.

Trouble shooting software is in many ways the same. It involves walking through the program. It is the changing program parameters to check a particular function. And it means using sophisticated tools to solve problems.

One of the best tools I have seen is a sharp program called **SOLVE™**. SOLVE is an acronym for Symbolic Object/Logic Verification and Examination. It allows software to be examined, tested and debugged. With it memory can be displayed, altered, and moved around. It also will assemble and disassemble code. Symbols and expressions can be used for labels and variables. It has 6 basic modes. They are:

```
1. Monitor commands
2. Assembler
3. Disassembler
4. Environment commands
5. Execution commands
6. Miscellaneous
```

It would impossible to cover everything it does. To do that I would be writing a manual and it already has fine one with it. But I will tell you a little about them and illustrate some applications.

Under Monitor Commands are:

```
M - Display memory
C - Examine and change memory
F - Fill memory
? - Search for bytes
X - Transfer memory
= - Calculate expression
```

I find that many times I will want to alter a module. Device descriptors are a good example. Say I have a descriptor D0. It has a stepping rate of 30 mS. This is is much slower than I would prefer. I want it to be 3 mS. So I enter SOLVE. I find D0 with:

```
DBG: L D0
87ED 87 .
```

At offset $14 (of the device descriptor)is the byte for disk speed. It is $00 which is a speed of 30 mS. I want to alter this to $03 to give me a speed of 6 mS. So I will use the change command.

```
DBG: C $87ED+$14
8801 00 .
```

My cursor rests just after the dot. I have a few options available to me. With a + or - I can move forward and backward through memory. A simple carriage return will permit me to exit back to debugger command prompt. An = followed by some expression will move me to a new location in memory. But I prefer to change this byte. So I enter a space which says that I am changing this location. So the next sequence appears:

```
8801 00 . ?$03
8802 20
DBG:
```

Notice the question mark. This is the prompt to enter a byte replacing the one that is already there.

If I want to make a permanent version of this I would use the following sequence from OS-9.

```
OS9: save /d0/modules/d0 d0
OS9: verify u </d0/modules/d0
       >/d0/modules/d0.temp
OS9: del /d0/modules/d0
OS9: rename d0.temp d0
```

This series of steps save D0 to a directory called MODULES, where other modules are stored. It corrects the CRC of the module creating a new one call D0.TEMP. The original is deleted and the temporary one is renamed to the original name. Now I can use OS9GEN to create a new system disk with the faster stepping rate.

Next is the Assembler. It has a number of directives. Some of them are:

```
A - Assemble command
R - Read lable file
W - Write label file
P - Print symbol table
```

It understands psuedo opcodes like ORG, OS9, and EQU. It has a single pass assembler that anticipates all the labels and symbols that have been assigned. It is not a full blown assembler, but it is handy for changing code on the fly.

A small example is in order. I have an area in memory which is reserved. I won't go into how this area was obtained, but it is there. From solve, I enter.

```
DBG:C $A000
A000 00 . " HELLO THERE!
A00D 00 . $ 0D
A00E 00 .
```

This little sequence plants "HELLO THERE!" starting at $A000 and terminates it with an end-of-line character. Now I have stored my message. I will create some code to print it.

```
DBG:A $A100
A100 WRITE LDX #$A000
A103  LDY #$0E
A107  LDA #01
A109 I$WRIT FN $8A
A109  OS9 I$WRIT
A10C  RTS
A10D
```

The last line has no entry. A carriage returns terminate the entry procedure and assembles the code. I haven't touched on how to execute the code. That will be coming.

One of my favorite features is the disassembler. I find many uses for it. Usually when I come across an OS-9 module of interest, I will use Solve's disassembler to see what makes it tick. Here is an example of using it on the code from before.

```
DBG:D $A100 $A10D
A100 WRITE LDX #$A000
A103 LDY #$0E
A104  LDA #$01
A109  OS9 I$WRIT
A10C  RTS
```

Notice how the code looks very similar with a few differences. It prints all the numbers in hex. And it did not include the line where I use 'FN'. This line is not a part of the assembled code, but rather it tells SOLVE that I$WRIT is equivalent to $8A. Besides disassembling, it also has a history function. Entering an 'H' will display the last 32 instructions that were executed. They are displayed disassembled when the trace mode or program simulation is being used.

Now comes the environment commands. These are used to alter the current conditions that influence a program. They include:

```
V - Define memory variables
: - Examine/change user stack
B - Set and display break points
K - Kill break points
^ - Print stack contents
@ - Change current nesting level
N - Set maximum nest tracking
level
```

I frequently use breakpoints when debugging programs. They provide a means to stop execution at a particular point. Imagine a runaway program. I have had a few of them in my time. I entered the modules name and it would load from the disk. And then nothing! Just silence as the computer sat there while it was stuck in an endless loop or somewhere in memory where no man has gone before. Many times things did happen. The printer would start spewing out nonsense. Strange graphics would appear on the video screen. The disk drives would whirl and click. Whatever was happening it was not good.

With breakpoints, the code can be executed in parts. It is a good idea to have a listing available. It will show the code and its offsets. I want to run a program called MYCODE and stop it before it runs amok. Perhaps the location I want to stop it at is $0900 relative to the start of the module. I would pre-load it into memory and then enter SOLVE. I would use the following sequence.

```
DBG: L MYCODE
A000 87 .
DBG: B $A000+$0900
A900
```

These lines link to MYCODE and set a breakpoint at $A900. This is the actual location, I want to stop. Now when I execute it, it will stop at $A900. If I enter B by itself, a listing of all the breakpoints would be displayed. Using the K command will remove any or all of them.

To be able to run the module from SOLVE requires the Execution Commands. They are:

```
L - Link to module
E - Prepare module for execution
T - Trace instructions
G - Run program
S - Simulate program
```

I find the L command to be handy for linking to modules that I want to examine. It gets the address. For Level 2, it maps the module into SOLVE's memory area.

The E command is a little more useful, when actual execution is desired. It also links to the module, but it also sets up the stack area. This means setting up the registers for memory requirements, parameter area, direct page and some other things. It is worthwhile to note that module and its data area are placed in SOLVE's memory area. This isolates it from the system and other processes.

G is used to start things again. Whatever is in register PC is executed . What is in PC is a result of using E, a breakpoint, a keyboard interrupt, or what has been place there from SOLVE.

Finally is the miscellaneous commands. They include:

```
$ - Pass command to OS-9
Q - Quit SOLVE
! - Set program base
< - Set data base
```

Remember before how I linked to a module and got its location. Well, I could have set the program base. Then everything I did could be offset from the base location. As an example, look at the breakpoint setting.

```
DBG: L MYCODE
A000 87 .
DBG: !$A000
A000
DBG: B !$0900
A900
```

This is like before, but the line where !$A000 occurs sets the base to $A000. Now when I set a breakpoint using the ! I can reference it to the way it appears in the code. Anything else I do with the code can be handled using the offsets. The < is used for setting a base for the module's data area.

SOLVE is a super debugger. There is so much more that I haven't covered. If I tried to cover it all, I would be writing a manual. That has already been done and it does an excellent job of covering things. I find that not a week goes by when I don't use it. I have to say this is one fine debugger. ( And the price ain't bad either! )

## CHECKING THE PATHS

There is an interesting command in UNIX. It is called FILE. When FILE is entered and a path follows it, it returns a message whether it is a directory or file. It is not a profound command, but it has its uses.

I decided to try my hand at writing one also. If the convention of creating directory names in upper case and files in lower case is followed, this command should not be necessary. Unfortunately this is not the case. I have received disks where everything was in uppercase characters. I have seen others where the case is mixed. Therefore, this month I offer a program called PATH.

PATH takes a pathname and analyses. It prints where it is a directory or a file. It also prints if it is a an SCF or RBF device. I have to admit there are a few definciencies with it. The major one is that it assumes whatever is not RBF must be SCF. This excludes Pipes and SBF ( Sequential Block File ) found in tape drives. But taking this into account, I believe you'll find it helpful. Please, improve on it if you wish.

Well another month has come to an end. Come back next time!

**LISTING**

```
0001 /* ****************************
0002 *
0003 *     Name: PATH.C
0004 *     Date: 23-NOV-87
0005 *     Author: R. D. Voigts
0006 *     To compile: cc path.c
0007 *
0008 *  ****************************
0009 *
0010 *     Function:
0011 *     Returns the status of a path
0012 *     indicting whether it is a file
0013 *     or directory and whether it is
0014 *     a SCF or RBF type device.
0015 *
0016 *  ****************************
0017 *
0018 *     History:
0019 *     V01.00  11/24/87  RDV
0020 *     Orignial.
0021 *
0022 *  **************************** */
0023 *
0024 *     Usage:
0025 *     OS9: path <path> [...]
0026 *
0027 *  **************************** */
0028 *
0029
0030 #include <stdio.h>
0031 #define DIR 128
0032
0033 main( argc, argv )
0034 int argc;
0035 char **argv;
0036 {
0037      register int i=0;
0038      int j;
0039      int err;
0040
0041 /* Put in help for program */
0042      if ( argc==1 )
0043           help();
0044
0045 /* Process the pathnames */
0046      while ( ++i<argc ) {
0047           j=access( argv[i], DIR ); /*
Test for directory */
0048           err=errno; /* Save the error
*/
0049           printf("\nPath: %s\n",
argv[i]);
0050           if ( !j )
0051                printf("Type:
Directory\n");
0052           else if ( err==203 )
0053                printf("Type:  SCF
Device\n");
0054           else if ( err==214 )
0055                printf("Type: File\n");
0056           else if ( err==221 )
0057                printf("Unknown
Device\n");
0058           else if ( err==216 )
0059                printf("Bad Path\n");
0060           else
0061                prerr( 1, err );
0062      }
0063 }
0064
0065 /* Help routine */
0066 help()
0067 {
0068      printf("Syntax: path <path>
[...]");
0069      printf("Usage:  Returns information
about the path.");
0070      exit( 0 );
0071 }
```

**EOF**

# Pascal

## A Tutorial

By: Robert D. Reimiller
Certified Software Corp
616 Camino Caballo
Nipomo, CA 93444
805 929-1359

This month we will take a look at an example of a VERY simple stand-alone program. This program has the task of reading the output of a 12 bit signed Analog to Digital (A to D) convertor every 10ms and updating a Digital to Analog (D to A) convertor with this value. The routine slows down the response of D to A convertor by only allowing it's value to change a small amount every 10ms. The input range of the analog voltage in the test setup was between -2000 and 2000, within the plus and minus 2047 available from a 12 bit convertor. If the A to D changes from one extreme to the other, the D to A convertor will take 4 seconds to make the full change. This can be calculated from :

```
range / ((samples/second) * (maximum step/sample))

In this example:
4000 / (100 * 10) = 4000 / (1000) = 4 seconds
```

This type of routine was used on a model train control system to prevent those who held the speed control (the A to D input) from making radical changes in the voltage to the train (the D to A output). Since I have done this, I can tell you it destroys the gears!

The timer used is a MC68230 Parallel Interface/Timer (PIT). Looking at the listing, the structure of the PIT is defined on lines 6 through 21. Record fields d1-d26 are dummy bytes, since the PIT is a byte wide device and the hardware is designed so the registers only show up on odd addresses, the even addresses must be skipped. The registers we are concerned with are the TCR (Timer control Register), CPRH-CPRL (Counter preload high-low), and the TSR (Timer status register).

On line 22 the PIT we will be using (which is located on the CPU board, so we call it CPUPIT) is located at $e0001, the compiler calculates the correct address of all of the registers in the PIT. The D to A output number 0 is an integer (16 bit signed) located at $ff0008 and has a valid range of -2048 to 2047 and is given the name DATA0. The ATOD_STATUS is a 16 bit value located at $ff0000, the most significant bit is one when there is data available. ATOD_DATA is an integer located at $ff0002 with a range of -2048 to 2047. ATOD_CHANNEL has the input channel as the lower 3 bits, and a strobe bit as the fourth bit, which starts a conversion.

At line 31 we start the actual interrupt handler, the interrupt being generated by the 68230. Note that while in this procedure we compile with debugging information off, this avoids the unpleasant situation of the debugger stopping in this section of code, which it is not designed to do. This procedure is declared as an exception procedure, which means that it has all the special stack handling required to be called as a 68000 series exception handler.

The first thing the RTCIRQ must do is to clear the source of the interrupt, this is done by setting a bit in the TSR PIT register. It then makes sure that the conversion is complete by checking the most significant bit of ATOD_STATUS and waiting until it is a one. Since this A/D takes considerably less than 10ms to make the conversion, this step is really not necessary, but I'd rather be safe than sorry. Line 39 reads the value from the A to D and saves it in VAL. Line 40 is used to start

the next conversion by setting in the channel number and strobe bit into ATOD_CHANNEL. Lines 43 through 51 take care of making sure that the new value for the D to A is no more than MAXSTEP from the previous value, or the actual difference, whichever is smaller, with the new value being written into the D to A on line 52. Lines 53 through 55 show how you can make delay counters using a simple interrupt source. These delays can be set and checked in the main program to control when things happen.

In the main program Lines 60-63 setup the necessary control registers for the timer so that it interrupts every 10ms. Line 65 starts an A to D conversion so that when the first interrupt occurs there will be valid data. Line 68 is inline assembly language code (signified by the ! at the start of the line) which is used to enable CPU interrupts by setting the interrupt mask to zero. Under the target debugger this is not actually necessary since it has already enabled interrupts so the link to the host system can work. Line 69 enables the timer to generates interrupts. The main part of the program would go where line 71 is.

It is fairly easy to get a target program to work. For any amount of serious development work, it is best to get the target debugger to work, this is a program similar to the host debugger that was described in a previous month. Instead of loading the object code into the host computer, the target debugger loads code into the target system RAM via a serial link. A special set of EPROMS is burned to go into the target system which controls the serial link, and has the debug kernal for communicating with the host. This normally involves selecting the driver for the type of serial chip used on the target, modifying the addresses, vector numbers, etc., and burning the EPROMS. Drivers for your hardware may be available from computer manufacturers that sell OmegaSoft Pascal, but if they aren't, someone familiar with the specific target hardware and assembly language can fairly easily make the modifications necessary. After this is done, the target debugger works similar to the host debugger, such as allowing breakpoints, variable display and change, and instruction tracing.

In order to setup the shell file for a target project, you need to use the linkage creator, very similar in use to what was shown for a OS-9 program in an earlier month. Instead of telling the linkage creator that you want "auto" setup, you answer no and you are then prompted for different information. For the example hardware, RAM went from location 8 through 7ffff (locations 8 through 3ff are for exception vectors). EPROM space started at location 80000. The PIT on the CPU board interrupted using autovector number 5 (vector 29). In the linkage creator :

```
Ram start (heap start) : 400
Ram end (stack upper limit) : 7ffff
Parameter list address (enter 0 if none) : 0
Maximum vector number to initialize : 255
Vectors in RAM ? Y
Enter vector (low[-high] label : 29 rtcirq
Enter vector (low[-high] label :
Starting load location : 80000
Library directory name : /dd/
Use default I/O library ,/dd/os9io, ? N
I/O library to use : testio
```

The rest of it is the same as for a program to run under OS-9. The I/O library in this case is very simple, it simply defines an error handler entry point (which should never be called in this example) and end of the varib storage section, such as :

```
testio    idnt      1,0
          xdef      .error,varibend
.error    move      #$2700,sr
          move.l    0,a7
          move.l    4,a0
          jmp       (a0)
          varib
varibend  equ       *
          end
```

The stack setup file has all the code needed to transfer vectors from EPROM in RAM during startup. Any vectors not defined to be handled by the Pascal program or a device driver are set to restart the program by default. To run the program under the target debugger there is only one command needed in filer mode to set the exception vector :

```
<F> SV V29 rtcirq
```

This tells the debugger to use the procedure rtcirq as the exception handler for vector number 29.

Next time we will start at the beginning, with basic data types, for both the 68020 version of OmegaSoft Pascal, and for an enhanced version of Modula-2 planned for later this year.

OmegaSoft is a registered trademark of Certified Software Corporation, OS-9 and OS-9/68000 are trademarks of Microware Systems Corporation.

+++

```
 1:0 program taat ;
 2:1   const
 3:2     channel = 1 ; {A/D channel number}
 4:2     atrobe = 8 ; {A/D atrobe bit}
 5:1   type
 6:2     pit = record
 7:2             pgcr, d1, parr, d2, paddr, d3, pbddr, d4, pcddr, d5,
 8:2             pivr, d6, pacr, d7, pbcr, d8, padr, d9, pbdr, d10,
 9:2             pasr, d11, pbsr, d12, pcdr, d13, psr, d14, null1, d15,
10:2             null2, d16 : byte ;
11:2             tcr {timer control regiatar} : byte ;
12:2             d17, tivr, d18, null3, d19 : byte ;
13:2             cprh {counter preload high} : byte ;
14:2             d20 : byte ;
15:2             cprm {counter preload mid} : byte ;
16:2             d21 : byte ;
17:2             cprl {counter preload low} : byte ;
18:2             d22, null4, d23, cntrh, d24,
19:2             cntrm, d25, cntrl, d26 : byte ;
20:2             tar {timer atatua regiater} : byte ;
21:2           end ;
22:1   var
23:2     cpupit : pit at $e0001 ;
24:2     data0 {amoothed output voltage} : integer at $ff0008 ;
25:2     atod_atatua : hex at $ff0000 ;
26:2     atod_data : integer at $ff0002 ;
27:2     atod_channel : hex at $ff0004 ;
28:2     output_value, delay : integer ;
29:2
30:2 {$d-}
31:1   procedure rtcirq ; exception  ;
32:2     conat
33:3       maxatep = 10 ;
34:2     var
35:3       val : integer ;
36*2     begin
37*3       cpupit.tar := 1 ; {clear interrupt}
38*3       while atod_atatua and $8000 <> 0 do ; {wait till ready}
39*3       val := atod_data ; {read A to D convertor}
40*3       atod_channel := channel + atrobe ; {atrobe it}
41:3       {only allow a maximum amount of
42:3        change from current output value}
43*3       val := val - output_value ;
44*3       if abs(val) > maxatep
45:4         then
46*5           if val < 0
47:6             then
48*7               val := -maxatep
49:6             alae
50*7               val := maxatap ;
51*3       output_value := output_value + val ;
52*3       data0 := output_valua ;
53*3       if delay <> 0
54:4         then
55*5           delay := dalay - 1
56*2     end ;
57:2 {$d+}
58:2
59*1   begin
60*2     cpupit.tcr := $a0 ; {aetup cOntrol regieter, interrupts off}
61*2     cpupit.cprh := 0 ;
62*2     cpupit.cprm := $9 ;
63*2     cpupit.cprl := $c4 ; {generata 10ma interrupt}
64*2     output_value := 0 ;
65*2     atod_channel := channel + atrobe ;
66*2     data0 := 0 ;
67*2     delay := 0 ;
68:2     ! move #$2000,ar anable interrupta ;
69*2     cpupit.tcr := $a1 ; {anable timer interrupta}
70*2     repeat
71:2       { main loop, delay can be used for 1ma delay }
72*2     until falae
73*1   end.
No Compilation Errora, atack = 0000001A aymbol tabla left = 47.2K
```

# Logically Speaking

Most of you will remember Bob from his series of letters on XBASIC. If you like it or want more, let Bob or us know. We want to give you - *what you want!*

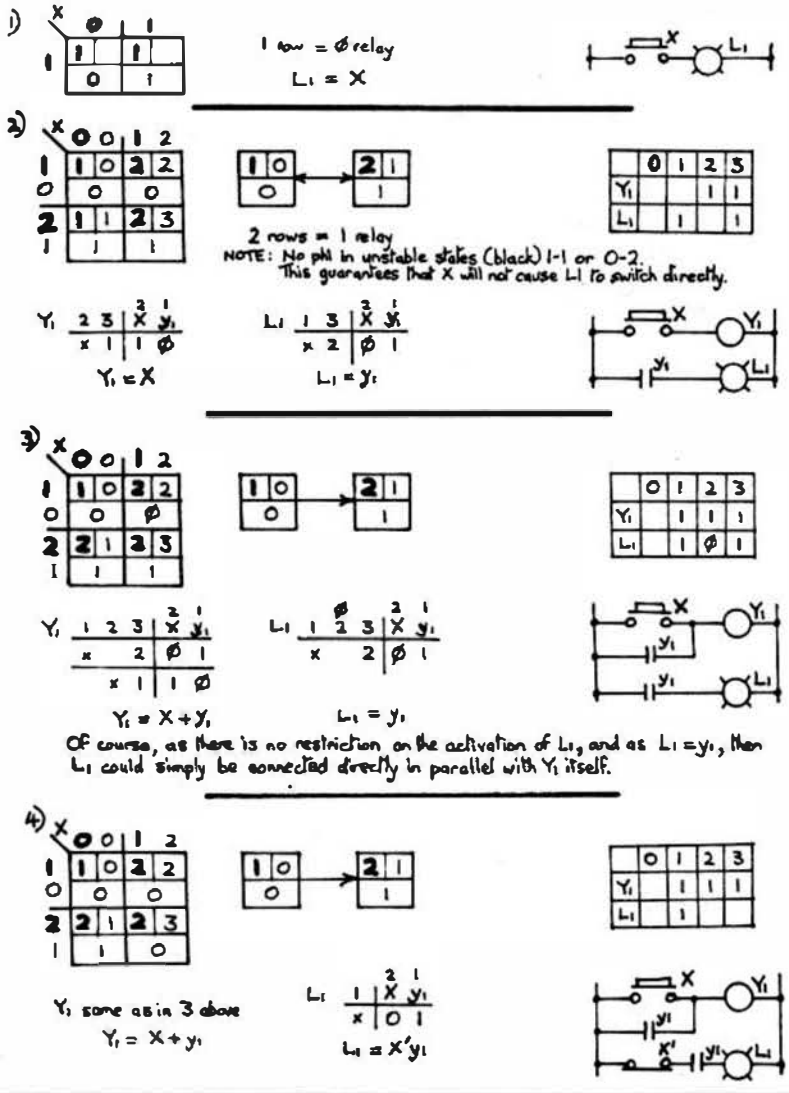
## The Mathematical Design of Digital Control Circuits

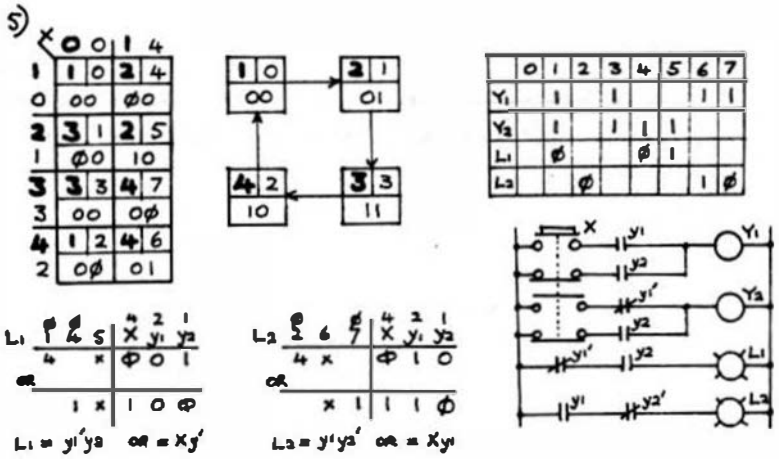
By: R. Jones
Micronics Research Corp.
33383 Lynn Ave., Abbotsford, B.C.
Canada V2S 1E2


Let me say again that when decoding relays, the minterms (red numbers) are selected by (a) examining all Box Cs of the state-diagram to see when a particular relay is energised, (b) noting the black number in Box A of any such Box C, and then (c) scanning the flow-table for corresponding black numbers in Box A. The red numbers at these locations specify the minterms.

Thus, in problem 2, Y1 is energised (has a 1 in Box C) with a matching black 2 in Box A. The flow-table has black 2s in Column black-1, corresponding to red-2 and red-3. Problem 3, on the other hand, has a black-2 in three of its Box-As, corresponding to red-1, 2 and 3.

When decoding for output devices, simply scan all Box Cs of the FLOW-TABLE itself, and pick off all red numbers corresponding to an energised or phi state for that device.





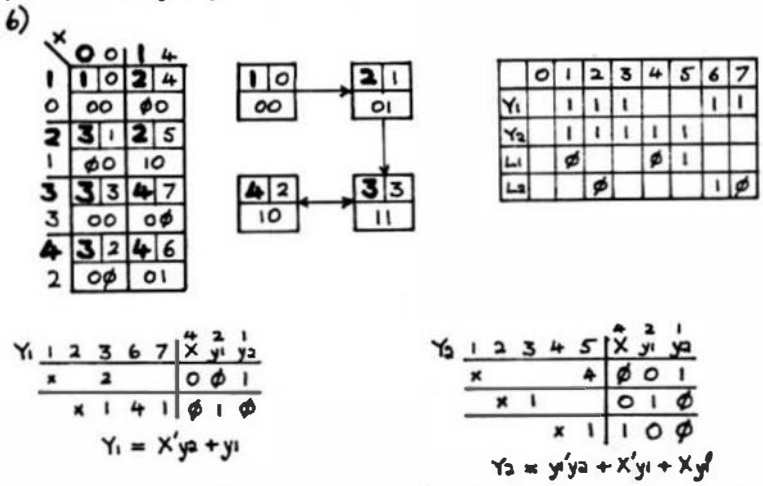
Because the state-diagram is exactly the same as that of Diagram 18 of an earlier lesson, the RELAY control circuitry will also be the same. The decodings appear in Diagrams 21 and 22, viz

Y1 = Xy1 + Xy2 and Y2 = Xy1' + Xy2

Note that the decoding for L1 begins with an 'x' under minterm-5, which is the first absolute minterm. 1 and 4 are both phis, so we use them only if necessary to optimise our decoding. Similarly with L2, which commences at minterm-6. Note too that there are two possible decodings for each light. Taking L1 as an example, we begin with minterm-5 and ask "Is 5 - 4 = 1 available? Yes!", so the 1 in column 'X' gets changed to a phi and a 4 is placed under minterm-1. (Why?) Then "Is 5 + 2 = 7 available? No! So how about 5 - 1 = 4? Yes, but we can't complete the run because 1 - 1 = 0 is not available."

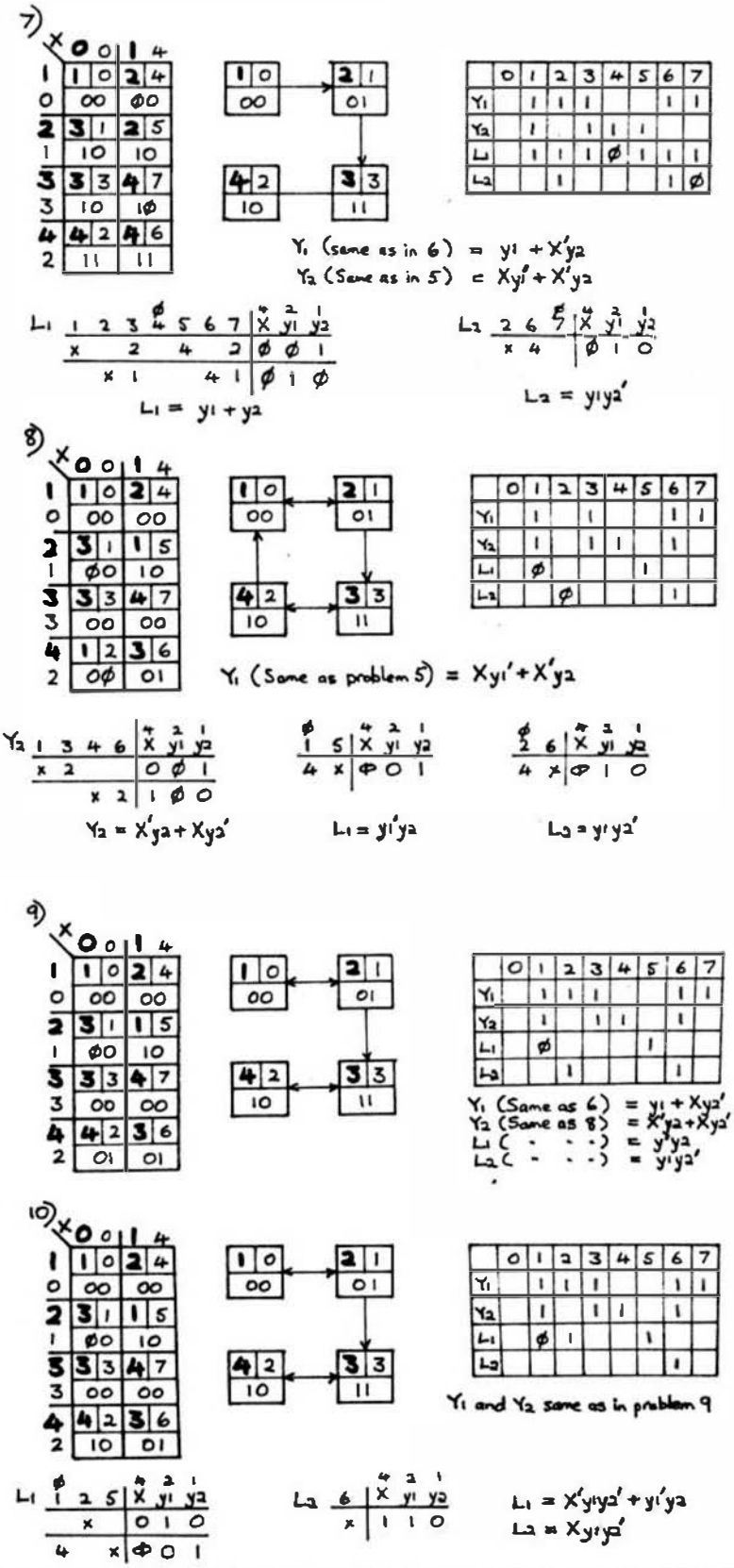
We'll elaborate on all this later, when we move on to a more comprehensive system, but for now we'll just try an alternative decoding, and begin by checking off the bit-position which blocked us in our first attempt. That is, we'll put an 'x' under minterm-5 as before, but begin by asking "Is 5 - 1 = 4 available? Yes, so we'll check off minterm-4, and change the 1 in column y2 to a phi." Then we'll go to the highest bit-position, and ask "Is 5 - 4 = 1 available? Yes, but 4 - 4 = 0 isn't, so we can't complete this run. Also 5 + 2 = 7 is not there, so we're all done!"

Sometimes the alternative decoding is useful in case the remainder of the circuitry has used up all the available contacts on a particular relay or push-button.



Note, in this example, that because the state-diagram is different, the relay controls will be different. However, the outputs, L1 and L2, repeat the pattern of Problem 6, and the preliminary table shows that their decodings will therefore be the same. I'd like to draw your attention to the fact that Sections A of the last two rows of the flow-table are identical (ie, 3,4) and to recommend that you try this problem once more when you've mastered the subject of MERGING, which we'll deal with in the next stages of our journey, where we'll be learning an enhanced technique for designing sequential circuits. This will help you to appreciate to the full the importance of MERGING, and the extra phis it creates for us!

As of now, I'll leave the drawing of the circuit-diagrams to you. I figure that by this time you should experience little difficulty once you've decoded the Boolean algebraic expressions for the various devices.



**Mile 7 - heading for Mile 8**

After all that, I've almost lost track of where we were. Ah, yes! I remember - we were going to let Uncle Fred tell us how he escaped from the M'bul-yans. Over to you, Uncle Fred!!

**UNCLE FRED'S STORY**

Well, like I said earlier, I knew that if I could only map my situation I'd got it solved! Problem was, how to map it! Very difficult - very, VERY difficult when you're as emotionally involved in it as I was!! Anyway, I decided to draw a little K-map so (here he scratches Diagram 25 in the dust), where S equals 'statement' and D equals 'death' (shudder!)

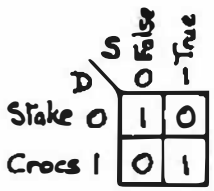



*Diagram 25*

In square 00 (Statement is false, death equals stake) I put a 1 to indicate that this was POSSIBLE. In square 10 (Statement is true, death equals stake) I put a 0 to indicate IMpossible. And so on for Row 1. So ... there was my situation all mapped out, but how was that going to help me formulate a statement for I-asku? Then it hit me!! I decided I'd rather create an IMpossible situation than a possible one, so I just picked one of the 0-squares, the one at location 10, and read off the co-ordinates, just as with any regular K-map. This translated as "The statement is true (that) I shall be burned at the stake" which I shortened to a more positive "I SHALL be burned at the stake!" Of course, I could just as easily have chosen the other 0-square at location 01, and read it out as "The statement is false (that) I shall be thrown to the crocs" which would shorten to "I shall NOT be thrown to the crocs!". And that did the trick! Back to you, Uncle Bob, and thanks for indirectly saving my life!!

**REMEMBER THE ABOVE! YOUR LIFE could DEPEND ON IT ONE DAY!**

Can you see why such a simple remark put I-asku in such a predicament? If not, try considering things from the ancestors' viewpoint if I-asku were to carry out either of the only two forms of execution allowed.

Uncle Fred, I think you should contact Aunt Minnie as soon as possible before she spends ALL your 95 dollars, though I have a feeling you may be too late, as you've been "lost" for several days, so let's get back to

**THE SYNTHESIS OF SEQUENTIAL CONTROL-CIRCUITS - AN ENHANCED TECHNIQUE**

The rules we've learned in the previous Lesson cover the basics of the design process, and having had a fair amount of practice with the problems of TEST 7, we're ready now for a refined technique, and perhaps a slightly more complex problem. So, without further ado, let's assume that we've just received the following set of specifications from a customer :

The machine is to consist of three lights, L1, L2 and L3 (all initially OFF), and two pushbuttons, X1 and X2, interlocked so that they cannot be operated simultaneously - - ie, the condition X1.X2 = 11 is IMpossible. Each time X1 is operated AND released, the lights are to come ON in the order L1 only (first press and release), L2 only (second press and release) and L3 only (third press and release). Once L3 is ON, subsequent operations of X1 to have no further effect. However, if X2 is operated and then released, the lights will move back by one position, and do this cyclically. That is, commencing with (say) L3, successive operations of X2 will cause it to cycle in the order L3 only, L2 only, L1 only, all OFF, L3, L2, L1 etc etc. The push-buttons may, of course, be operated in any random order.

**THE FLOW-TABLE**

The first step, as before, is to draw up a flow-table (see Diagram 26) with the primary-control columns headed 00, 10 and 01. 11 is not included as this represents an impossible phi-state. At this stage we don't know how many rows we're going to need, so we'll just add them on as we go along!

| $X_1X_2$ | 00 | 10 | 01 |
|---|---|---|---|
| 1 | 1 | 2 | 8 |
| | 000 | 000 | 000 |
| 2 | 3 | 2 | |
| | ϕ00 | 000 | |
| 3 | 3 | 4 | 9 |
| | 100 | 100 | 100 |
| 4 | 5 | 4 | |
| | ϕϕ0 | 100 | |
| 5 | 5 | 6 | 10 |
| | 010 | 010 | 010 |
| 6 | 7 | 6 | |
| | 0ϕϕ | 010 | |
| 7 | 7 | 7 | 11 |
| | 001 | 001 | 001 |
| 8 | 7 | | 8 |
| | 00ϕ | | 000 |
| 9 | 1 | | 9 |
| | ϕ00 | | 100 |
| 10 | 3 | | 10 |
| | ϕϕ0 | | 010 |
| 11 | 5 | | 11 |
| | 0ϕϕ | | 001 |

*Diagram 26*

Initially the machine is at rest, with all lights OFF, so address 00.1 is allocated a black-1 to maintain stability, and an all-zero entry in Box-C to keep all lights OFF. The first OPERATION of X1 takes us to address 10.2, with all lights still OFF, and the first RELEASE of X1 then moves us to address 00.3, which is a stable state with L1 alone ON. Note that because L1 alone is changing state while it transits via address 00.2, this address specifies L1 as a phi. We'll decide later whether to activate it right here, or leave it till a split-second later at address 00.3, depending on how the decoding turns out.

The second OPERATION of X1 takes us as far as address 10.4, with the outputs held constant at 100 in order to maintain L1 ON and the other two OFF, and the second RELEASE moves us to address 00.5 (stable), with an output entry of 010 to switch L1 OFF and turn L2 ON. In Box-C of address 00.4 we enter a phi for both lights L1 and L2, as they are BOTH changing state across this "elbow".

The third OPERATION of X1 brings us to address 10.6, with L2 maintained ON, and the third RELEASE moves us to address 00.7, with L3 alone ON, and an entry of phi for L2 and L3 in the elbow address 00.6, as both are changing state here.

Keeping in mind that we're not to proceed beyond L3 in the "UP" direction, so the fourth operation of X1 simply keeps us in Row-7 with L3 held ON. Obviously, any further OPERATIONs or RELEASEs of X1 cannot move us out of this row, and we are stuck there with L3 ON and the other two OFF. All the action so far is contained within the heavy lines on the flow-table.

**NOW FOR X2**

So much for X1. Now let's transfer our attention to X2, and, commencing at address 00.1 (all lights OFF), assume that we OPERATE X2. We know from the specs that we should come to rest with L3 ON when we later RELEASE X2, so it would seem natural to insert a black-7 in the elbow 01.1 and then move to address 01.7, so that when X2 is released we would be in address 00.7 with L3 ON - - exactly as specified.

BUT, Row-7 would then be a completely stable row (with black-7 in all its Sections A), LEAVING US WITH ABSOLUTELY NO WAY TO MOVE OUT. This would be OK if the specs called for a termination at L3 no matter in which direction we cycled the lights, but this is not so in our case! We must therefore be a little more subtle and move into address 00.7 by the back-door as it were, and leave the front-door (at address 01.7) open as an emergency exit.

The way to do this is to send ourselves down to address 01.8 when we OPERATE X2 (still keeping all lights OFF) and move into 00.7 via the elbow at 00.8, with L3 a phi on the elbow itself, as it's changing state here. How about that for a crafty manoeuvre?

Next we'll assume that we're at address 00.3 (with L1 ON) when we decide to OPERATE X2. We'd like to move from the elbow 01.3 to address 01.1, so that when we RELEASE X2 we'd end up at 00.1 with all lights OFF. Unfortunately, address 01.1 specifies that L1 goes OFF here, where we wish to maintain it ON. In addition, its Box-A is coded to send us instead to a final "L3 only ON" state, so we adopt the same tactics as before and "sneak in" to 00.1 via row 9. Our elbow 01.3 will therefore have a black-9 entered in Box-A and we'll end up at address 01.9 with L1 still ON. When we RELEASE X2, the entry of a black-1 in 00.9 will send us up to 00.1 and a corresponding all-zero condition for the lights. The elbow 00.9 will, of course, have a phi entry for L1.

Similarly, commencing at 00.5 (L2 alone ON), we cycle through Row 10 into address 00.3 (L1 alone ON), and finally, commencing at our problem-row 7 (L3 alone ON), we cycle through Row 11 into 00.5. Note that both of these movements call for a double-phi entry in the elbow-address, or unstable location. At this point we've successfully translated the specs into a flow-table which accurately follows the sequences called for.

*IT CANNOT BE TOO STRONGLY STRESSED THAT IF ANY DOUBT EXISTS, AT ANY TIME, AS TO THE CYCLING ON THE FLOW-TABLE, AN EXTRA ROW SHOULD BE CREATED AND THE ACTION CYCLED THROUGH THIS ROW.* The next stage of the synthesis procedure, which I'm afraid we'll have to leave till the next leg of our journey, will AUTOMATICALLY eliminate any surplus, or redundant, rows which you might so create.

As it stands right now, with 11 rows, the flow-table indicates that we can implement this circuit with four relays. Three isn't enough, because 2 to the power of 3 equals 8, and we could only cover 8 rows. On the other hand, 2 to the power of 4 equals 16, which would leave 5 unused rows in a 16-row table. This means 5 rows of phi-states, and we know how we love those little phis, don't we? In addition, we've already got a whole phi-column, namely column 11, PLUS a random assortment of phi-states (addresses into which the flow-table doesn't cycle) in columns 10 and 01, which should help us considerably in our decoding.

First though, we'll take a look at MERGING, which I mentioned earlier, to see whether it's possible to reduce the number of relays, or, if not, maybe we can at least create a lot more phis to play with.

Unfortunately, I've used up my allocation of space for this month, so we can all take a much-needed break till next time round. No tests for you this time, I'm afraid, but don't worry too much about that. Maybe we'll make up for it later!!

... End of Mile 7

*Correction to solutions to test four (4):*

(viii) should read: a'bc'+bc'd+a'd+b'c

**EOF**

## UTILITIES

Basic09 XRef from S.E. Media -- This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.
*O & CCO obj. only -- $39.95; w/ Source - $79.95*

BTree Routines - Complete set of routines to allow simple implementation of keyed files - *for your programs* - running under Basic09. A real time saver and should be a part of every serious programmers tool-box.
*O & CCO obj. only - $89.95*

Lucidata PASCAL UTILITIES (Requires Pascal ver 3)

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE -- Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.
*F, S, CCF --- EACH 5" - $40.00, 8" - $50.00*

DUB from S.E. Media -- A UniFLEX BASIC decompiler Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works w/ ALL Versions of 6809 UniFLEX basic.
*U - $219.95*

LOW COST PROGRAM KITS from Southeast Media The following kits are available for FLEX, SK*DOS on either 5" or 8" Disk.

1. **BASIC TOOL-CHEST $29.95**
   BLISTER.CMD: pretty printer
   LINEXREF.BAS: line cross-referencer
   REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS: remove superfluous code
   STRIP.BAS: superfluous line-numbers stripper
2. **FLEX, SK*DOS UTILITIES KIT $39.99**
   CATS. CMD: alphabetically-sorted directory listing
   CATD.CMD: date-sorted directory listing
   COPYSORT.CMD: file copy, alphabetically
   COPYDATE.CMD: file copy, by date-order
   FILEDATE.CMD: change file creation date
   INFO.CMD (& INFOGMX.CMD): tells disk attributes & contents
   RELINK.CMD (& RELINK82): re-orders fragmented free chain
   RESQ.CMD: undeletes (recovers) a deleted file
   SECTORS.CMD: show sector order in free chain
   XL.CMD: super text lister
3. **ASSEMBLERS/DISASSEMBLERS UTILITIES $39.95**
   LINEFEED.CMD: 'modularise' disassembler output
   MATH.CMD: decimal, hex, binary, octal conversions & tables
   SKIP.CMD: column stripper
4. **WORD - PROCESSOR SUPPORT UTILITIES $49.95**
   FULLSTOP.CMD: checks for capitalization
   BSTYCIT.BAS (.BAC): Stylo to dot-matrix printer
   NECPRINT.CMD: Stylo to dot-matrix printer filter code
5. **UTILITIES FOR INDEXING $49.95**
   MENU.BAS: selects required program from list below
   INDEX.BAC: word index
   PHRASES.BAC: phrase index
   CONTENT.BAC: table of contents
   INDXSORT.BAC: fast alphabetic sort routine
   FORMATER.BAC: produces a 2-column formatted index
   APPEND.BAC: append any number of files
   CHAR.BIN: line reader

**BASIC09 TOOLS** consist of 21 subroutines for Basic09. 6 were written in C Language and the remainder in assembly. All the routines are compiled down to native machine code which makes them fast and compact.

1. CFILL -- fills a string with characters
2. DPEEK -- Double peek
3. DPOKE -- Double poke
4. FPOS -- Current file position
5. FSIZE -- File size
6. FTRIM -- removes leading spaces from a string
7. GETPR -- returns the current process ID
8. GETOPT -- gets 32 byte option section
9. GETUSR -- gets the user ID
10. GTIME -- gets the time
11. INSERT -- insert a string into another
12. LOWER -- converts a string into lowercase
13. READY -- Checks for available input
14. SETPRIOR -- changes a process priority
15. SETUSR -- changes the user ID
16. SETOPT -- set 32 byte option packet
17. STIME -- sets the time
18. SPACE -- adds spaces to a string
19. SWAP -- swaps any two variables
20. SYSCALL -- system call
21. UPPER -- converts a string to uppercase

For OS-9 - $44.95 - Includes Source Code

See Review in January 1987 issue of 68 Micro Journal

## SOFTOOLS

The following programs are included in object form for immediate application. PL/9 source code available for customization.

READ.ME Complete instructions for initial set-up and operation. Can even be printed out with the included text processor.
CONFIG one time system configuration.
CHANGE changes words, characters, etc. globally to any text type file.
CLEANTXT converts text files to standard FLEX, SK*DOS files.
COMMON compare two text files and reports differences.
COMPARE another check file that reports mis-matched lines.
CONCAT similar to FLEX, SK*DOS append but can also list files to screen.
DOCUMENT for PL/9 source files. Very useful in examining parameter passing aspects of procedures.
ECHO echos to either screen or file.
FIND an improve find command with "pattern" matching and wildcards. Very useful.
HEX dumps files in both hex and ASCII.
INCLUDE a file copy program that will accept "includes" of other disk files.
KWIC allows rotating each word, on each line to the beginning. Very useful in a sort program, etc.
LISTDIR a directory listing program. Not super, but better than CAT.
MEMSORT a high-speed text file sorter. Up to 10 fields may be sorted. Very fast. Very useful.
MULTICOL width of page, number of columns may be specified. A MUST!
PAGE similar to LIST but allows for a page header, page width and depth. Adjust for CRT screen or printer as set up by CONFIG. A very smart print driver. Allows printer control commands.
REMOVE a fast file deleter. Careful, no prompts issued. Zap, and its gone!
SCREEN a screen listing utility. Word wraps text to fit screen. Screen depth may be altered at run time.
SORT a super version of MEMSORT. Ascending/descending order, up to 10 keys, case over-ride, sort on nth word and sort on characters if file is small enough, sorts in RAM. If large file, sort is constrained to size of your largest disk capacity.
TPROC a small but nice text formatter. This is a complete formatter and has functions not found in other formatters.
TRANSLIT sorts a file by x keyfields. Checks for duplications. Up to 10 key files may be used.
UNROTATE used with KWIC this program reads an input file and unfolds it a line at a time. If the file has been sorted each word will be presented in sequence.
WC a word count utility. Can count words, characters or lines.

**NOTE: this set of utilities consists of 6 5-1/4" disks or 2 8" disks, w/ source (PL/9). 3 5-1/4" disks or 1 8" disk w/o source.**
**Complete set SPECIAL INTRO PRICE:**
5-1/4" w/source FLEX - SK*DOS - $129.95
w/o source - $79.95
8" w/source - $79.95 - w/o source $49.95

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants -- TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.
*F, S and CCF, U - $25.00, w/ Source - $50.00*

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No 'blind' debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!
*Levels I & II only - OS-9 $69.95*

## DISK UTILITIES

OS-9 VDisk from S.E. Media -- For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.
*Level I OS-9 obj. $79.95; w/ Source $149.95*

O-F from S.E. Media -- Written in BASIC09 (with Source), includes: REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX, SK*DOS Format so it can be used normally by FLEX, SK*DOS; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX, SK*DOS Directory, Delete FLEX, SK*DOS Files, Copy both directions, etc. FLEX, SK*DOS users use the special disk just like any other FLEX, SK*DOS disk
*O - 6809/68000 $79.95*

I.SORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. I.SORT includes a full

set of comments and errors messages.
*OS-9 $85.00*

HIER from S.E. Media - *HIER is a modern hierarchal storage system for users under FLEX, SK*DOS.* It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. **Using HIER a regular (any) FLEX, SK*DOS** disk (8 - 5 - hard disk) can have sub directories. By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. **Each directory looks to FLEX, SK*DOS like a regular file, except they have the extension '.DIR'.** A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX, SK*DOS. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!
*FLEX - SK*DOS $79.95*

COPYMULT from S.E. Media – Copy LARGE Disks to several smaller disks. FLEX, SK*DOS utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fooling with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.
*Completely documented Assembly Language Source files included.*
*ALL 4 Programs (FLEX, SK*DOS, 8" or 5") $99.50*

COPYCAT from Lucidata – *Pascal NOT required.* Allows reading TSC Mini-FLEX, SK*DOS, SSB DOS68, and Digital Research CP/M Disks while operating under SK*DOS , FLEX1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.
*F, S and CCF 5" - $50.00 F, S 8" - $65.00*

VIRTUAL TERMINAL from S.E. Media - Allows one terminal to do the work of several. The user may start as many as eig t task on one terminal, under *VIRTUAL TERMINAL*, and switch back and forth between task at will. No need to exit each one; just jump back and forth. Complete with configuration program. The best way to keep up with those background programs.
O & CCO - obj. only - $49.95

FLEX, SK*DOS DISK UTILITIES from Computer Systems Consultants -- Eight (8) different Assembly Language (w/ Source Code) FLEX, SK*DOS Utilities for every FLEX, SK*DOS Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). – PLUS – Ten XBASIC Programs including: A BASIC Resequencer with EXTRAs over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.
*ALL Utilities include Source≥ (either BASIC or A.L. Source Code).*
*F, S and CCF - $50.00*
*BASIC Utilities ONLY for UniFLEX -- $30.00*

## COMMUNICATIONS

CMODEM Telecommunications Program from Computer Systems Consultants, Inc. – Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".
*FLEX, SK*DOS, CCF, OS-9, UniFLEX, 68000 & 6809 Source $100.00 - without Source $50.00*

X-TALK from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eig t inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk with C-MODEM (68020). Text and binary files may be directly transferred between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.
*X-TALK Complete (cable, 2 disks) $99.95*

*X-TALK Software (2 disks only) $69.95*
*X-TALK with CMODEM Source $149.95*

XDATA from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc.
*U - $299.99*

## EDITORS & WORD PROCESSING

JUST from S.E. Media – Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with PAT or any other editor.

* Now supplied as a two disk set:
*Disk #1: JUST2.CMD object file,*
*JUST2.TXT PL9 source:FLEX, SK*DOS - CC*
*Disk #2: JUSTSC object and source in C:*
*FLEX, SK*DOS - OS9 - CC*

The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p .u .y etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!
*Disk (1) - PL9 FLEX only- F, S & CCF - $49.95*
*Disk Set (2) - F, S & CCF & OS9 (C version) - $69.95*
*OS-9 68K000 complete with Source - $79.95*

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.
*Regular FLEX, SK*DOS $129.50*
** SPECIAL INTRODUCTION OFFER * $79.95*
*SPECIAL PAT/JUST COMBO (w/source)*
*FLEX, SK*DOS $99.95*
*OS-9 68K Version $229.00*
*SPECIAL PAT/JUST COMBO 68K $249.00*
Note: *JUST in "C" source available for OS-9*

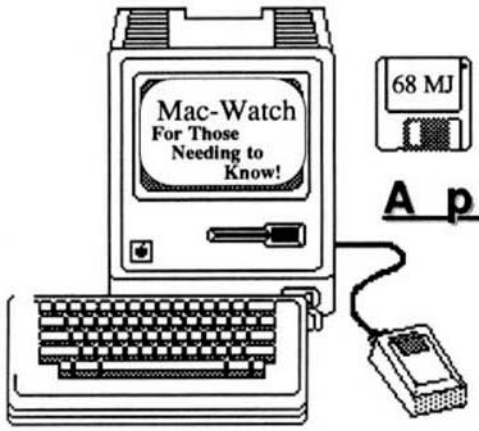
CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassel' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - appx. 14,000 plus of free memory! Extra fine for programming as well as text.
*FLEX, SK*DOS $69.95*

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entires lines, etc. Complete with over 25 different CRT terminal configuration overlays.
*FLEX, CCF, SK*DOS $39.95*

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!
*6800 or 6809 FLEX, SK*DOS or SSB DOS, OS-9 - $175.00*

SPELLB "Computer Dictionary" from S.E. Media – OVER 150,000 words! *Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX, SK*DOS UCS).* Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.
*F, S and CCF - $129.95*

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR – (uses the 51 x 24 Display Screens on CoCo FLEX/SK*DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.




## *The Macintosh™ Section*

**Reserved as a**

# A place for your thoughts

***And ours.....***

# Mac-Watch

## Spellswell™ Revisited

### A Spelling Checker & Proofreader

Last year we ran a review of this same program. Since then it has been enhanced to reflect the needs of more recent Macintosh word processing applications. As many of our readers who rely on our reviews are more "serious" users, we thought it appropriate to bring things up to date. This brings us up to version 2.0e, which is current as of the end of 1987.

The following is a partial listing of current applications that this version functions with, preserving graphics, font and style information.

```
MacWrite 4.5, 4.6
Word*    3.01
Works
More
ThinkTank
Jazz
Acta
Text documents
* Will not work with Word
"Fast Save" files.
```

Included is a "homonym" checker, allowing for comparison of words that have similar sounds. Example "to - two - too." Examples of each word is displayed and how each word is used. Users can add words to the "homonym dictionary" for special treatments.

Another nice feature is correction of abbreviations that are not entered correctly. Example "Phd - PHD - Ph.D." For the technical writer this feature has significant value.

It has a 93,000 plus word main dictionary. And the user can create and maintain a separate dictionary of special words that are not appropriate for the main dictionary.

Other features include: automatic word replace or skip functions, recognize proper nouns that are not capitalized, uncapitalized words at the start of any sentence and incorrect hyphenated words. It also detects missing apostrophes and missing spaces between sentences and words. For the chronic hyphenater, it recognizes words such as "pre-sorted" to be two words, such as "pre sorted" and treats each as a separate word.

The user has the option of making a new document of the one being checked and changed. This document will be the mirror image of the original unaltered document and has appended the extension ".sbk." Words that contain diacritics may be added to the dictionaries.

All the quick features are supported and are optionally selected and may be saved as permanent options. These may be changed during a session as a temporary function, or saved permanently. The quick functions are, Skip, Replace, Add and Delete. Questioned words are shown in context. Also A "Guess" option allows the checker to suggest a proper spelling. Wildcard guesses are allowed by the insertion of a "?" question mark. For instance, the word "simultaneously" may be wildcarded as "sim?ly." All words starting with "sim" and ending with "ly" will be displayed. Needless to say, this function can become quite time consuming, but can sure make life easier for the user who does a

lot of word processing. Help is online and selected from the dialog box. The View option allows the user to view, in alphabetical order, listings direct from the selected dictionary. This is sorta like using a standard printed dictionary, without the finger walking.

Other useful features include checking for double words, one or two spaces after a period and words that are made up of mixed letters and numbers. Words connected by double dashes are treated as separate words. Spellswell allows for the insertion of abbreviations, special capitalization and contractions into the dictionaries. Most of the options can be turned off, as the user desires.

Spellswell is a "batch" as opposed to an "interactive" spelling checker. That is, you run it as an application and choose each document as a separate run. Interactive spelling checkers can run as you type in your document and normally have two modes, "interactive" or "selection batch spelling checking." I actually use both. I use the "interactive" checker as I compose documents. However, many of our articles, advertising and other textual material comes in over modem, or is on a non-Macintosh disk and has to be ported to the Macintosh format for editing and processing. In that case we always use Spellswell, after it is ported to the Macintosh, as it is undoubtedly the best for that type of application.

There is no limit to the size of documents this checker can handle. On exiting the Spellswell session a report of the number of words checked and total word count is displayed. Also the user is given the option of saving any special options (skips, etc.) selected for that particular session for any future spelling checks of the same document.

Additional special dictionaries are available from the vendors of Spellswell. They are a legal and medical dictionary. It should be noted that these are additional cost items and are not included, as is the case with some other spelling checkers. As we did not receive them they are not included in this review.

## Lookup™ and Findswell™

These two utilities are DAs (Desk Accessories) from the folks that produce Spellswell. Once you start using either or both, you are hooked. Even if you use similar programs from other vendors, these two are tops in what they do.

### Lookup

This is a utility, for 512K and up Macintoshes, that uses the dictionaries from Spellswell and allows the lookup of any word, from within any application. It is certainly much slicker than having to stop and look up words out of heavy and bulky printed dictionaries. Not to mention much faster (I don't care how fast your fingers can walk through the pages!) Lookup comes with the standard 93,000 plus word dictionary. Same one as is used by Spellswell. Using Lookup allows checking any word with just a simple keystroke.

Lookup makes spelling suggestions and then will replace the word in question with the correct spelling. Also supported is the wildcard feature of Spellswell. Words may be entered or deleted from the dictionaries and Lookup maintains the format and font of the word being changed. Also featured is an excellent Help function, again practically the same as for Spellswell but directed towards the specifics of Lookup.

The dialog box has several options, including Sound - checked the system beeps twice for functions successfully completed, unchecked the sound is shut off. Also supported are the options covered in the Spellswell review above, including View, Guess, Add, Delete and Replace, as well as Cancel, Capitalization, Abbreviations, contractions and diacritical marks are all handled as in the Spellswell review above.

Lookup is the sort of program that becomes more indispensable as it is used. At first I was calling up an "inside" spelling checker on practically every editing session. That was taking up a lot of valuable memory as well as slowing things down. Now I depend a lot on Lookup alone, as it is a DA and always there under the Apple menu. I don't even have a regular printed dictionary on my desk anymore. That should tell you something about the latest spelling aids now available for the Macintosh.

**Findswell**

This is one that I thought that I really didn't need, when it first arrived. Boy, was I wrong!

You see, the Macintosh now comes with a real neat DA called "Find File." Everyone who purchases a newer Macintosh receives the Find File DA as part of the utilities supplied from Apple. And I must admit that it was one of the most used DAs we had on our various systems (we have everything from 128K Macs to Mac IIs in our office), because of the hundreds and even thousands of files we maintain on our hard disks. Before we had Find File we would literally spend a good chunk of an hour sometimes looking for some file on our hard disks. The Find File allowed us to type in the name or part of the name of a file and it would be found in seconds, as opposed to minutes (many) before. We were happy with Find File, that is until we started using Findswell.

Findswell is an "Init" type file. An Init file is one that resides in the System Folder and as the system is booted up, each time, all Init files are located and executed before the system is turned over to the user. Sort of an automatic program installer.

Findswell has features that go beyond Find File. The one that we find most useful is it's ability to have the pointer relocated to the resident folder of any file we search for. With Find File it was necessary, after the path to the file was determined, to point your way through the various levels of folders until you arrived at the proper one, or let Find File move the located document to the desktop until you completed your work on that particular document. First, we don't like to work from the desktop. Secondly, if you have a hard disk, or even a floppy, that has many levels of folders, it can get to be a real bear finding your way around. You can do a lot of mouse punching.

As I said above, Findswell, once it has located a document (Findswell actually finds anything on a disk - document or application, etc.), positions the selection pointer in the proper folder for immediate opening. We have some files that are as far as 9 to 10 folders or more deep. As you can see, a lot of time can be saved by automatic pointer location.

Findswell, once placed in the system folder, inserts an additional box in the "Open" dialog box. When Findswell is selected another dialog box is opened where you type all or part of the name of the document desired. Options are - All, Full Name, First Part, Stop, Open and Done. When the located document is displayed at the top of the dialog box the entire path as well as the full name is shown, date and time of its last modification, its size and the program that created it. The document can then be opened from the open button or the box closed and the application Open function activated and the pointer is immediately in the proper folder.

Documents that you frequently use and folder names can be remembered by Findswell. Each time you use Findswell these remembered names will appear and can be opened with the normal double click.

For those of you who do not have Find File (older Macintosh) I would certainly recommend considering Findswell. And, if you use a hard disk as heavily as we do, then I would recommend Findswell, even if you already have Find File.

A staff review.

**EOF**

# Atari Users Corner

*The Place Where Contributors Meet*

## ATARI ST 68000 COMPUTERS

Dale E. Randall
1270 Dew Drop Lane
Florissant, MO. 63031

### INTRODUCTION

Like many other computer hobbyists, my first home computer experience was with a SWTPC 6800 kit. I purchased it in 1976. This machine started out as three memory board, 12K machine. I used a Micro-Term ACT I, TV terminal which displayed 16 lines by 64 uppercase characters. I made my own 300 Baud "Kansas City" tape cassette interface. The 8K Basic took at least ten minutes to load. I developed my own 6800 Editor/Assembler and a powerful but slow 12K Basic interpreter. This was all done, using a Radio Shack stereo tape deck. I even had a Teletype KSR 35 with punched tape reader and tape punch. I also developed my own 256x256 bit graphics 8K memory board. I redesigned the tape cassette interface to work at 2400 baud. The terminal and TV set was replaced with a Digital Research ZRT 80 CRT Terminal board and Zenith green screen. The Teletype was replaced by a uni-directional Centronics 737 dot matrix printer. This eventually evolved into a full 48K system, which needed a 10 amp power supply to avoid a "brown out", when our home central air conditioner turned on. The SWTPC 6800 machine was finally retired and replaced with a HELIX 6809, with FLEX, in the fall of 1983.

### THE HELIX

The HELIX was a vast improvement. It had 256K of bank switched DRAM memory, a 35 amp regulated power supply, a 360K floppy disk drive with DMA, a 6809 2.5 MHZ processor, plus compatibility with the old SWTPC machine. The extra memory above 56K became a RAM disk. By now, as you probably have suspected, my hobby is with both hardware and software. As an engineer, I could have afforded to buy everything already designed and built, but I have a lot of fun doing my own interfacing and writing the software for it.

The HELIX, with it's compatible S64 and S30 buses, was "enhanced" with some of the old S50 and S30 boards from the SWTPC system. I designed a homemade A/D joystick board and a software controlled baud rate board. The printer was replaced with an Epson FX-80 bi-directional dot matrix printer. Later, I purchased and adapted the COCO OS9 level 1, to work in the Helix, which also used the extra memory for a RAM disk. I upgraded the 256K memory board to 1 Megabyte. Two more half size floppy drives were added. I wrote my own eprom monitor. The B command automatically "boots" either the FLEX or OS9 system disks. In either case, the entire operating system, with all of the utility commands, are loaded into the RAM disk. This allows commands to be almost "instantaneous". This system was soon "supplemented" with an ATARI 520ST 68000 system in the fall of 1985.

### THE ATARI ST

The 520ST had 512K of memory, a color RGB analog monitor, TOS in RAM, a GEM Desktop, a two button mouse with extra joystick port, a 64K cartridge slot, an RS-232 serial interface, an IBM parallel printer interface, a MIDI interface, hard disk interface, and a single sided 360K 3.5" external disk drive. This was called a "Color Mac". The system was supplied with Logo, Basic, First Word, and Neochrome and the price was under $1000! Soon the TOS in ROM became available for $30, and was installed. This 192K operating system now boots up in a few seconds, without even needing a floppy disk. Many command shells are available. This allows us old timers to use either the standard mouse or the old familiar MS-DOS command line.

### IBM MS-DOS CAPABILTY

Now the amazing thing, that I found out, was that the ST uses the MS-DOS disk directory structure. It is "IBM compatible". I soon added an external 5 1/4" 360K drive. I spliced an ATARI 14 pin cable to a standard 34 pin flat lead cable and added it as drive B:. The "select" pull-up resistor had to be disconnected so that the ST could select it. My drive requires a "poke" to change the drive step rate from 3 to 6 milliseconds. I now have a machine that allows me to take IBM floppy disks home from work. I can edit IBM ASCII files at home. Both the IBM and ST store CR/LF characters at the end of each line. They both also allow TAB characters to be used. We wrote a FORMAT utility, that eliminated the need to format the disks on the IBM.

Soon, with the aid of ATARI's programmers development package, I was able to port my own Editor and Basic interpreter over from the HELIX to the ST, via RS-232. I have expanded the ST memory to 1 meg. I added an ATARI SH204 hard disk drive. The drive is really a standard 20 meg with an Adaptec controller. I partitioned it into C:, D:, and E: directories. The PC-DITTO program which emulates the IBM PC/XT 8088 system, allows most of the IBM software disks to be inserted, and run directly on the ST. It even allows the user to boot up drive C: as an XT, and drive D: as an ST. The emulation speed, with the 68000, is about half as fast as a PC/XT, but the best of both worlds are usable on one machine. The ST easily emulates the IBM color or monochrome boards with its 80 column, low resolution, 200 by 320, eight color system. The ST has more capability with eight levels for each color.

### CPM 80 CAPABILITY

The GENIE bulletin board system, that is available in all major cities, has very low evening rates. There are over 8000 public domain CPM 80 files that are available on this system. There is a public domain CPM 80 emulator for the ST, that runs as fast as an actual 2 MHZ Z80 system.

### MACINTOSH CAPABILTY

There is now available a "Magic Sac" cartridge that plugs into any 520ST, 1040ST, MEGA ST2, or ST4 cartridge slot. It allows the user to throw away his old Mac and use the Mac ROMs in the ST. It runs most of the Macintosh programs, with a wider screen and graphics resolution. The increased ST memory allows the Macintosh operating system software to be put in RAM disk. There is a dramatic improvement in speed, because the system software doesn't have to reside on the user's disk. MAC owners are amazed to see their old familiar software run on the ST, without any modifications.

A hardware interface allows the ST floppy disk to operate with either a it's Macintosh type disk controller, or with the standard 1772 controller in the ST. This eliminates the need to convert Macintosh formatted disks to ST formatted disks via RS-232.

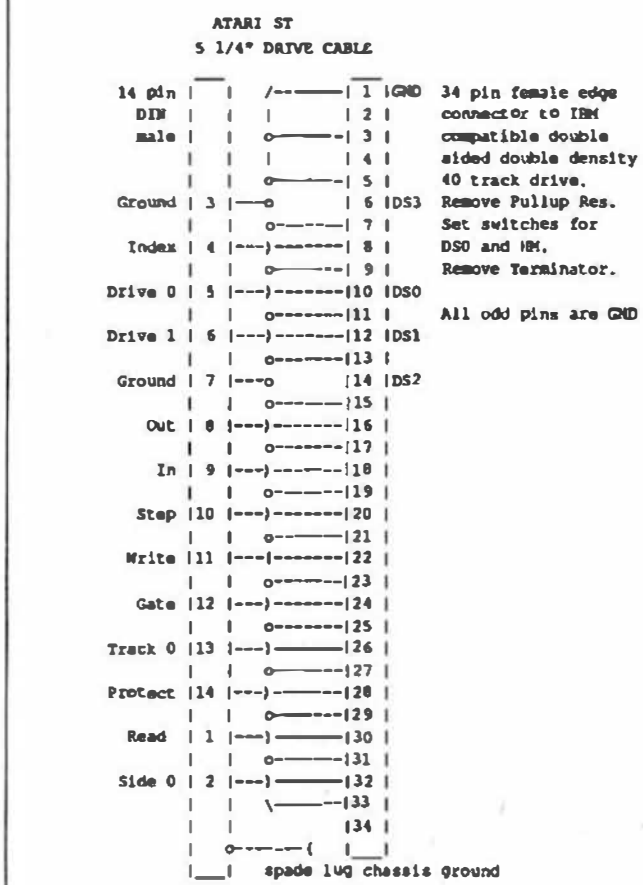
### ST SOFTWARE AVAILABILITY

Many thousands of public domain files are free for the cost of the time to download them from bulletin board systems. GENIE has over 5000 ST files in it's libraries. Our St. Louis ATARI computer store, has over 800 programs that can be ordered and received in less than three days. I counted over 200 different programs "on the shelf", for the ST and Mega ST. Many of these are games, but over one half of them are not:

```
 5 Accounting
35Adventure Games
 3 Computer Emulators
 2 Databases
 2 Desk Top Publishing
18Educational
30Graphic Arts
 5 Home Management
 5 Language Compilers
60Misc. Games
 3 Modem Terminal Emulators
 5 Music Composer Players
 2 Shell Debuggers
15Sports Games
 3 Spreadsheets
 4 Text Editors
 3 Word Processors
```

### CONCLUSION

I encourage ST users to submit and share their ideas in this magazine. Included are some utility programs, that are written for the ST. I submit them as public domain software for others ST enthusiasts to use or modify as needed. I believe that the new ATARI MEGA ST, which has the faster "Blitter" and it's affordable cost, will become a common 68000 machine. The ST has already done this, both in North America and in Europe.




```
/*   Colored Hat Drawing Program      */
/*          by Jeff Randall           */
/*      Public Domain Software        */

#include "osbind.h"

extern float sqrt();
extern float sin();

int  contrl[12],intin[128],ptsin[128],
     intout[128],ptsout[128],handle,
     whand,chstat,wchar,alldone,pcolor,
     asca,oldpal[16],xy[4];

/* Define NEO Information */

int header[2] = {0x0000,0x0000};
int data[46] =
{
0x2020,0x2020,0x2020,0x2020,0x2e20,0x2020,0x801e,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
};

/* Define Colors */

int palette[16] =
{
0x0000,0x0700,0x0730,0x0750,
0x0770,0x0470,0x0070,0x0075,
0x0077,0x0057,0x0027,0x0007,
0x0507,0x0707,0x0704,0x0702
};

static int cofset[16] =
{
0,2,3,6,4,7,5,8,9,10,11,14,12,15,13,1
};

double sqin,sqout;

long fhand,lchar,slphys;

char decimal[20];

/* Main processing routine */

main()
{
float getflt();

int  i,l_intin[11],l_out[57],
     gr_1,gr_2,gr_3,gr_4,l_ptsin[20],
     a,b,count,climit,cdivfac,p,q,x1,x2,xl,xi,xp,y1,y2,yp,zi,zp,zz;

float  xf,xr,xt,xx,yf,yr,yy,zf,zt;

/* Start the program! */

appl_init();

handle=graf_handle(&gr_1,&gr_2,&gr_3,&gr_4);

/* open workstation */

for (i = 0; i < 10; i++)
```

```
  l_intin[i] = 1;
l_intin[10] = 2;
v_opnvwk(l_intin, &handle, l_out);
v_hide_c(handle);
for (i=0; i<16; i++)
  oldpal[i] = setcolor(i,-1);
slphys=physbase();
vsm_type(handle,1);

/* main program loop */

alldone=0;
while(alldone == 0)
  {
  v_clrwk(handle);
  setpallete(oldpal);

/* Input Hat parameters */

  crlf();
  prompt("P=");
  p=getflt();
  prompt("Q=");
  q=getflt();
  prompt("XP=");
  xp=getflt();
  prompt("YP=");
  yp=getflt();
  prompt("ZP=");
  zp=getflt();
  prompt("Resolution (1 to 10) ");
  a=getflt();

  xr=1.5*3.141593;
  yr=1;
  xf=xr/xp;
  yf=yp/yr;
  zf=xr/zp;
  xl=9999;

  v_clrwk(handle);
  setpallette(palette);

/* Process the Pi ture */

  for(zi=0-q; zi<=q-1; zi=zi+a)
    {
    if((zi>=0-zp) && (zi<=zp))
      {
      zt=zi*xp/zp;
      zz=zi;
      xl=sqrt(xp*xp-zt*zt)+.5;
      for(xi=0-xl; xi<=xl; xi=xi+a)
        {
        xt=sqrt(xi*xi+zt*zt)*xf;
        xx=xi;
        yy=(sin(xt)+.4*sin(3*xt))*yf;
        x2=xl;
        y2=yl;
        xl=xx+zz+p;
        yl=yy-zz+q;
        if(xl<x2)
          {
          x2=xl;
          y2=yl;
          }
        for(b=1-a; b<=a-1; b++)
          {
          pcolor=0;
          vsl_color(handle,cofset[pcolor]);
          xy[0]=xl+b; xy[1]=199-yl; xy[2]=xl+b; xy[3]=199;
v_pline(handle,2,xy);
          chstat=bconstat(2);
          if(chstat != 0)
            {
            b=a;
            xi=xl+1;
            zi=q;
            }
          }
        p olor=(1.4*yf+yy)/5;
        while(p  lor>14)
          pcolor=pcolor-15;
         vsl_color(handle,cofset[pcolor+1]);
         xy[0]=x2; xy[1]=199-y2; xy[2]=xl; xy[3]=199-yl;
v_plline(handle,2,xy);
        }
      }
    }

 /* Hat done, wait for key */

  chstat=0;
  while(chstat == 0)
    chstat=bconstat(2);
    lchar = bconin(2);
    wchar = lchar & 0x00FF;

/* Check for RETURN */

    if(wchar == 0x000d)
      {
      alldone=1;
      }
    else

/* Check for fun tion key F1 */

    if(lchar == 0x003b0000)
      {
      fhand=fcreate(&"colorhat.neo",0);
      if (fhand >= 0)
```

```
      {
      whand=fhand;

/* Write header (4 bytes) */

      fwrite(whand,4L,&header);

/* Write color palette (16 words) */

      fwrite(whand,32L,&palette);

/* Write neo data (92 bytes) */

      fwrite(whand,92L,&data);

/* Write picture data (32000 bytes) */

      fwrite(whand,32000L,slphys);
      fclose(whand);
      alldone=1;
      }
    }
  }

/* Close the workstation. */

v_clsvwk(handle);
setpallete(oldpal);

appl_exit();
_exit(0);

}

/* Print string w/ CR & LF */

print(string)
char *string;
{
cconws(string);
crlf();
}

/* Print string (no CR/LF) */

prompt(string)
char *string;
{
cconws(string);
}

/* Output CR/LF to screen */

crlf()
{
cconout(13);
cconout(10);
}

/* Accept Y/N response */

yesno()
{
int exit,yorn;

exit = -1;
while(exit < 0)
 {
 yorn = bconin(2);
 if ((yorn == 0x004e) || (yorn == 0x006e))
  {
  print("No");
  exit = 0;
  }
 else
 if ((yorn == 0x0059) || (yorn == 0x0079))
  {
  print("Yes");
  exit=1;
  }
 }
return(exit);
}

/* Accept floating-point number */

float getflt()
{
int i,getfx,inct,negct,decct,decfnd;
float work,mfac,innum;
static int fltwk[20];

cconout('?');
negct=0;
decct=0;
inct=0;
getfx=0;
while(getfx == 0)
 {
 chstat=0;
 while(chstat == 0)
  chstat=bconstat(2);
 lchar = bconin(2);
 wchar = lchar & 0x00FF;
 if((wchar == 0x0008) && (inct > 0))
  {
  cconout(8);
  cconout(32);
  cconout(8);
  inct--;
  if(fltwk[inct] == 98)
   decct=0;
  }
 else
 if(fltwk[inct] == 99)
  {
  negct=0;
  }
 }
else
if(wchar == 0x000d)
 {
 if((inct-decct-negct) > 0)
  {
  crlf();
  fltwk[inct]= -1;
  getfx=1;
  }
 }
if(inct < 19)
 {
 if((wchar > 0x002f) && (wchar <
0x003a))
  {
  fltwk[inct]=wchar-48;
  inct++;
  cconout(wchar);
  }
 else
 if((wchar == 0x002d) && (negct == 0)
&& (inct == 0))
  {
  negct++;
  fltwk[inct]=99;
  inct++;
  cconout(wchar);
  }
 else
 if((wchar == 0x002e) && (decct == 0))
  {
  decct++;
  fltwk[inct]=98;
  inct++;
  cconout(wchar);
  }
 }
}
decfnd=0;
innum=0;
mfac = 10;
for(i=0; i<inct; i++)
 {
 if(fltwk[i] == 98)
  {
  decfnd = 1;
  }
 else
```

```
  if(fltwk[i] < 10)
   {
   if(decfnd == 0)
    {
    innum=innum*10.0;
    innum=innum+fltwk[i];
    }
   else
    {
    work=fltwk[i];
    work=work/mfac;
    innum=innum+work;
    mfac=mfac * 10.0;
    }
   }
  }
if(negct == 1)
 innum = -innum;
return(innum);
}

+++


/* 40 Track Format Program */
/*          by             */
/*     Jeffrey Randall     */

#include "define.h"
#include "osbind.h"

int  contrl[12],intin[128],ptsin[128],intout[128],ptsout[128],
  error,errcnt,handle,gr_1,gr_2,gr_3,gr_4,i,l_intin[11],l_out[57],
    t,chr,type=1,flag=0;

long amount,filler,serno=1,magic=0x87654321;

char buffer[10000];

main()
{
  appl_init();
  handle = graf_handle(&gr_1,&gr_2,&gr_3,&gr_4);
  for(i=0;i<10;i++) l_intin[i]=1;
  l_intin[10]=2;
  v_opnvwk(l_intin,&handle,l_out);
  v_hide_c(handle);
  v_clrwk(handle);
  Cconout(27);
  Cconout(69);
  error = puts("        ************************");
  error = puts("        * Format 40  Ver 1.0 *");
  error = puts("        *  by  Jeff Randall  *");
  error = puts("        ************************");
  Cconout(10);
  error = puts("     This program will format an");
  error = puts("  IBM 40 track DSDD disk in drive B!");
  error = puts("         Press 'Y' to format");
  error = puts("      or any other key to abort!");
  while (Bconstat(2) != 0) chr=Bconin(2);
  chr=Bconin(2);
  if (chr == 'Y' | chr == 'y')
  {
    Cconout(13);
    Cconout(10);
    error = printf("                                          |");
    Cconout(13);
    for(t=0; (t<40) & (errcnt < 5); t++)
    {
      error = printf("#");
      error = 1;
      errcnt = 0;
      while((error != 0) & (errcnt < 5))
      {
        error = Flopfmt(&buffer,filler,1,9,t,0,1,magic,0xE5E5);
        errcnt = errcnt + 1;
      }
      if (errcnt < 5)
      {
        error = 1;
        errcnt = 0;
        while((error != 0) & (errcnt <5))
        {
          error = Flopfmt(&buffer,filler,1,9,t,1,1,magic,0xE5E5);
          errcnt = errcnt + 1;
        }
      }
    }
    for(t=0; t<512; t++) buffer[t] = 0;
    for(t=1; (t<10) & (errcnt < 5); t++)
    {
      error = 1;
      errcnt = 0;
      while((error != 0) & (errcnt < 5))
      {
        error = Flopwr(&buffer,filler,1,t,0,0,1);
        errcnt = errcnt + 1;
      }
    }
    for(t=1; (t<10) & (errcnt < 5); t++)
    {
      error = 1;
      errcnt = 0;
      while((error != 0) & (errcnt < 5))
      {
        error = Flopwr(&buffer,filler,1,t,0,1,1);
        errcnt = errcnt + 1;
      }
    }
    buffer[0] = 0xFD;
    buffer[1] = 0xFF;
    buffer[2] = 0xFF;
```

```
    if (errcnt < 5)
    {
      error = 1;
      errcnt = 0;
      while((error != 0) & (errcnt < 5))
      {
        error = Flopwr(&buffer,filler,1,2,0,0,1);
        errcnt = errcnt + 1;
      }
    }
    if (errcnt < 5)
    {
      error = 1;
      errcnt = 0;
      while((error != 0) & (errcnt < 5))
      {
        error = Flopwr(&buffer,filler,1,4,0,0,1);
        errcnt = errcnt + 1;
      }
    }
    buffer[0] = 0;
    buffer[1] = 0;
    buffer[2] = 0;
    Protobt(&buffer,serno,type,flag);
    if (errcnt < 5)
    {
      error = 1;
      errcnt = 0;
      while((error != 0) & (errcnt < 5))
      {
        error = Flopwr(&buffer,filler,1,1,0,0,1);
        errcnt = errcnt + 1;
      }
    }
    if (errcnt > 4)
    {
      printf("\nError #%d  Format ABORTED!",error);
    }
  }
  Cconout(13);
  Cconout(10);
  if (chr == 'Y' | chr == 'y')
  {
    error = puts("Press any key to return to desktop");
    chr=Bconin(2);
  }
  v_clsvwk(handle);
  appl_exit();
  _exit(0);
}

+++
```

```
TTL     SPOOL.S Spooler Program for ATARI ST
*
*               Public Domain
*
*       The SPOOL.TTP program accepts a parameter  number
* of 1 to 999.  This number is the number of Kbytes  that
* it reserves in memory to buffer all characters that are
* sent to the printer via the BIOS Bconout function.
* The default value for a zero value is 50 (K). Then  the
* program terminates and stays resident, until the system
* is rebooted. If the buffer becomes full and the printer
* is not ready for 30 seconds, then a  printer  not-ready
* status is returned.

PRN     EQU     0       PRINTER OUTPUT DEVICE
GEMDOS  EQU     1
BCONOUT EQU     3       CONSOLE CHAR OUTPUT
SETEXBC EQU     5       SET EXCEPTION VECTOR
CONSTAT EQU     8       CONSOLE OUTPUT STATUS
MFPINT  EQU     13      MFP INTERRUPT INSTALLED
BIOS    EQU     13      ST BIOS TRAP 13
XBIOS   EQU     14      ST XBIOS TRAP 14
ISRB    EQU     16      INTERRUPT SERVICE REG B
TIMEOUT EQU     30      30 SECONDS TIMEOUT
KEEP    EQU     $31     HOLD RESIDENT PROGRAM
SAVPTR  EQU     $4A2    BIOS SAVE AREA/REGISTER
HZ_200  EQU     $4BA    200 HZ COUNTER
MFP     EQU     $FFFA01 MFP 68901
PSG     EQU     $FF8800 PSG YM 2149

        .TEXT

SPOOL   MOVEA.L 4(A7),A0        GET BASE PAGE ADDRESS
        MOVE.L  #256,D6         BASE PAGE SIZE
        ADD.L   12(A0),D6       + TEXT SIZE
        ADD.L   20(A0),D6       + DATA SIZE
        ADD.L   28(A0),D6       + BSS SIZE
        MOVEQ   #0,D7
        MOVEQ   #0,D0
        LEA     129(A0),A0      PARAMETERS ADDRESS
NEXTCHR MOVE.B  (A0)+,D0        GET REQUESTED BUFFER SIZE
        SUBI.B  #'0',D0
        BMI     EXIT            NOT A NUMBER DIGIT
        CMPI.B  #9,D0
        BGT     EXIT
        MULU    #10,D7          GOOD DIGIT
        ADD.W   D0,D7
        CMP.L   #100,D7         3 DIGITS MAXIMUM
        BLO     NEXTCHR
EXIT    TST.W   D7              NUMBER NON-ZERO?
        BNE     OK
        MOVE.W  #50,D7          NO, USE 50K
OK      EXT.L   D7
        MOVEQ   #10,D0
        LSL.L   D0,D7           TIMES 1024 (1K)
```

```
        ADD.L   D7,D6           BYTES NEEDED
        MOVE.L  D7,LENGTH       ENTER IN IOREC
        MOVE.L  #TRAP13,-(A7)   VEC
        MOVE.W  #45,-(A7)       VECNUM FOR TRAP 13
        MOVE.W  #SETEXEC,-(A7)
        TRAP    #BIOS           SET VECTOR
        ADDQ.L  #8,A7
        MOVE.L  D0,TRAPSVE      KEEP OLD VECTOR
        MOVE.L  #BUSYINT,-(A7)
        MOVE.W  #0,-(A7)        INTERRUPT NUMBER
        MOVE.W  #MFPINT,-(A7)
        TRAP    #XBIOS          PRINTER INTERRUPT ENABLED
        ADDQ.L  #8,A7
        CLR.W   -(A7)
        MOVE.L  D6,-(A7)        NUMBER OF BYTES
        MOVE.W  #KEEP,-(A7)     TERMINATE AND STAY RESIDENT
        TRAP    #GEMDOS

*       NEW TRAP #13 ROUTINE

TRAP13  MOVEA.L A7,A2           MARK SSP
        BTST    #5,(A7)         SUPEVISOR CALL?
        BNE     SUPER
        MOVE    USP,A2          NO, USE USER STACK POINTER
        SUBQ.W  #6,A2
SUPER   CMPI.W  #8CONOUT,6(A2)  BCONOUT CALL?
        BNE     NORMAL
        CMPI.W  #PRN,8(A2)      PRINTER CALL?
        BNE     NORMAL
        MOVEA.L SAVPTR,A1
        MOVE.W  (A7)+,-(A1)     SAVE STATUS
        MOVE.L  (A7)+,-(A1)     SAVE RETURN ADDRESS
        MOVE.L  A1,SAVPTR       SAVE PTR UPDATES
        MOVE.W  10(A2),D1       GET CHAR
        BSR     PRINT
        MOVEA.L SAVPTR,A1
        MOVE.L  (A1)+,-(A7)
        MOVE.W  (A1)+,-(A7)
        MOVE.L  A1,SAVPTR
        RTE

NORMAL  CMPI.W  #CONSTAT,6(A2)  PRINTER STATUS?
        BNE     NORM1
        CMPI.W  #PRN,8(A2)
        BNE     NORM1
        MOVEQ   #-1,D0          STATUS OK
        BSR.W   GETPTR          GET POINTER
        MOVE.L  TAIL(A0),D2
        BSR     WRAP
        CMP.L   HEAD(A0),D2     ROOM IN BUFFER?
        BNE     ROOM            YES
        MOVEQ   #0,D0           NO, BUSY, NO ROOM
ROOM    RTE

NORM1   MOVEA.L TRAPSVE,A0      TO OLD TRAP #13
        JMP     (A0)

PRINT   MOVE    #$2700,SR       INTERRUPT BLOCK
        BSR     GETPTR          POINTER TO IOREC & MFP
        MOVE.L  TAIL(A0),D2
        CMP.L   HEAD(A0),D2     BUFFER EMPTY?
        BNE     INBUFF
        BTST    #0,(A1)         PRINTER BUSY
        BNE     INBUFP
NOTBUSY LEA     PSG,A2          PSG ADDRESS
        MOVE.B  #15,(A2)        PORT B
        MOVE.B  D1,2(A2)        OUTPUT A BYTE
        MOVE.B  #14,(A2)        PORT A
        MOVE.B  (A2),D0
        ANDI.B  #$DF,D0         STROBE LOW
        MOVE.B  D0,2(A2)
        ORI.B   #$20,D0         STROBE HIGH
        MOVE.B  D0,2(A2)
        MOVEQ   #-1,D0          OK
        RTS

INBUFF  MOVE.L  TAIL(A0),D2     INCREMENT WRITE POINTER
        BSR     WRAP
        CMP.L   HEAD(A0),D2     BUFFER FULL?
        BEQ     BUFFULL
INBUF1  MOVEA.L (A0),A1         NO, BUFFER ADDRESS
        MOVE.B  D1,(A1,D2.L)    WRITE CHAR TO BUFFER
        MOVE.L  D2,TAIL(A0)     NEW TAIL INDEX
        MOVEQ   #-1,D0          OK
        RTS

BUFFULL MOVE.L  HZ_200,D0       SECONDS TO WAIT
        ADDI.L  #200*TIMEOUT,D0
        MOVE    #$2300,SR       INTERRUPTS FREED UP
WAIT    CMP.L   HEAD(A0),D2     MORE ROOM IN BUFFER?
        BNE     INBUF1
        CMP.L   HZ_200,D0       NO, TIME UP YET?
        BHI     WAIT
        MOVEQ   #0,D0           NOT OK
        RTS

* INTERRUPT ROUTINE TO OUTPUT CHARS TO PRINTER

BUSYINT MOVEM.L D0-D2/A0-A2,-(A7)       SAVE REGISTERS
        BSR     GETPTR          GET POINTERS
        MOVE.L  HEAD(A0),D2
        CMP.L   TAIL(A0),D2     BUFFER EMPTY?
        BEQ     EMPTY
        BSR     WRAP            NO, BUMP READ POINTER
        MOVEA.L (A0),A2         GET BUFFER ADDRESS
        MOVE.B  (A2,D2.L),D1    GET CHAR FROM BUFFER
        BSR     NOTBUSY         OUTPUT TO PRINTER
        MOVE.L  D2,HEAD(A0)     NEW READ POSITION
EMPTY   BCLR    #0,ISRB(A1)     CLEAR SERVICE BIT
        MOVEM.L (A7)+,D0-D2/A0-A2       RESTORE
        RTE
```

```
GETPTR  LEA     IOREC,A0        BUFFER FILE RECORD PTR
        LEA     MFP,A1
        RTS

WRAP    ADDQ.L  #1,D2           POINTER TO NEXT POS.
        CMP.L   LEN(A0),D2      END OF BUFFER?
        BLO     NOWRAP
        MOVEQ   #0,D2           YES, START AT TOP
NOWRAP  RTS

        .DATA

IOREC   DC.L    BUF             BUFFER ADDRESS
LENGTH  DC.L    1               BUFFER SIZE
        DC.L    0               WRITE INDEX
        DC.L    0               READ INDEX

BUFFER  EQU     0               IOREC BUFFER ADDRESS
LEN     EQU     4               IOREC BUFFER LENGTH
HEAD    EQU     8               IOREC WRITE PTR
TAIL    EQU     12              IOREC READ PTR

        .BSS

TRAPSVE DS.L    1               TRAP#13 VECTOR SAVE
BUF     EQU     *               START OF BUFFER MEMORY

        .END
```

+++

```
    .TTL    SLOW.S  D.E.Randall  PUBLIC DOMAIN

    * SETS ST OR MEGA ST DRIVE B:
    * STEP RATE = 6 MILLISECONDS
    * PUT SLOW.PRG IN AUTO FOLDER

    GEMDOS  EQU     1       TRAP #1
    Super   EQU     32      GEMDOS SUPERVISOR MODE
    INMEGA  EQU     $A52
    INROM   EQU     $A0C
    INRAM   EQU     $6CE

        .TEXT

SLOW    MOVE.L  #STACK,SP       USER STACK AREA
        CLR.L   A6              A6 = 0
        BSR     SMODE           SUPERVISOR MODE
        CMP     #3,INMEGA(A6)   MEGA TOS IN ROM?
        BNE     NOT1
        CLR     INMEGA(A6)      6 MSEC STEP RATE
        BRA     EXIT            FOR DRIVE B

NOT1    CMP     #3,INROM(A6)    ST TOS IN ROM?
        BNE     NOT2
        CLR     INROM(A6)       6 MSEC STEP RATE
        BRA     EXIT            FOR DRIVE B

NOT2    CMP     #3,INRAM(A6)    ST TOS IN RAM?
        BNE     EXIT
        CLR     INRAM(A6)       6 MSEC STEP RATE

EXIT    BSR     UMODE           USER MODE
        CLR     -(SP)           Pterm0
        TRAP    #GEMDOS         RETURN TO SYSTEM

SMODE   CLR.L   D0
UMODE   MOVE.L  D0,-(SP)        SYSTEM STACK PTR
        MOVE    #Super,-(SP)
        TRAP    #GEMDOS         USER MODE
        ADD.L   #6,SP
        RTS

        .BSS

        DS      252
STACK:  DS      4

        .END
```

# Bit-Bucket

**By: All of us**

*"Contribute Nothing - Expect Nothing", DMW '86*

1708 Piedmont St.
Jackson, MS 39202
26 December 1987

Editor
68 Micro Journal
Hixson, TN 37343

Dear Mr. Williams:

Please find enclosed a 5" Flex disk SSSD, which contains a file named Index87.txt. This is the 1987 Key Word Index for 68 Micro Journal, which has now become an annual holiday tradition for me. As you will remember, this index uses key words to facilitate searches for a specific topic or article, using a utility such as Leo Taylor's Find.cmd. You may publish it, use it, or distribute it as you see fit. I hope you find it as useful as I have over the years.
I must confess to having been tempted by "other brands" this past year, but have not succumbed. I did upgrade my 6809 system to 2 mhz and added a ram disk from D. P. Johnson. Now my system is so blazingly fast that I guess I would have to get an AT with hard disk to equal its speed. It looks like I'll be a 6809 Flex single user for some time to come.

Sincerely,
John D. Current

JAN 87 P8 ANDERSON USER NOTES BURNOUT PROGRAM ORGANIZATION LLOYD IO ED CRACKER CRASMB
JAN 87 P12 PASS C USER NOTES PROPOSED ANSI STANDARD HEADER FILES MATHEMATICAL FUNCTIONS B+ TREE LISTING
JAN 87 P18 VOIGTS BASIC OS9 COLUMN RESERVING MEMORY RMB DEVICES KBASIC LISTING
JAN 87 P21 GROSS ARTICLE MOTOROLA LAYOUT DESIGN VLSI CRITICAL PATH METHOD CPM CAD
JAN 87 P25 VOIGTS DESCRIBES BASIC09 TOOLS OS9 FUNCTIONS COMPILER WARNINGS PARSE CHARACTER STRINGS C LISTING
MAR 87 P18 VOIGTS BASIC OS9 PIPES WORD WRAP AROUND WRAP.C LISTING
MAR 87 P22 MACINTOSH SECTION SPELLSWELL SPELLING CHECKER
MAR 87 P25 LURIE FORTH TUTORIAL ANSI GRAPHICS ESC SEQUENCES FORTH LISTING
MAR 87 P29 BALITSKI ARTICLE BUILD AN RS232 BREAKOUT BOX
MAR 87 P36 CURRENT ARTICLE 1986 KEY WORD INDEX TO 68 MICRO JOURNAL
MAR 87 P40 TAYLOR HIER UNIX LIKE UTILITIES CONT. C LISTINGS
MAR 87 P42 JONES LETTER TSC XBASIC MAX LINE LENGTH RANDOM FILES
MAR 87 P44 DREXLER ARTICLE REALTIME CLOCK FOR FLEX MSM5832 PIA TIME.CMD 6809 ASSEMBLY LISTING SCHEMATIC
APR 87 P8 PASS C USER NOTES ANSI C STANDARD IMPLEMENTATION SPECIFIC DEPENDENCIES ENUMERATED DATA TYPES C LISTING
APR 87 P14 VOIGTS BASIC OS9 BASIC09 PRINT USING DISK HIERARCHY AND INTEGRITY CHECKER HCHECK BASIC09 LISTING
APR 87 P19 STRAUB ARTICLE INTERFACING MC68881 FPCP WITH MC6809 CPU SCHEMATIC PL9 LISTING
APR 87 P25 LURIE FORTH TUTORIAL NULL MODEM FILE TRANSFER FORTH LISTING
APR 87 P28 WILLIAMS RAMBLINGS OS9 VERSIONS "PAKS" MUSTANG08 FLEX ON 68000 OS9 ARCHIVE
APR 87 P37 MACINTOSH SECTION BATTERY PAK DESK ACCESSORIES
APR 87 P38 STAFF REVIEW SIM68K 68000/68010 SIMULATOR FOR IBM PC
APR 87 P39 WELLER REVIEW XDMS DBMS DATA BASE MANAGEMENT WESTCHESTER APPLIED BUSINESS SYSTEMS
APR 87 P41 LAVOREL FLEX UTILITY LOC.CMD FIND A SEQUENCE OF BYTES IN BINARY FILE 6809 ASSEMBLY LISTING
APR 87 P45 TAYLOR HIER UNIX LIKE UTILITIES CONT. 6809 ASSEMBLY LISTING
APR 87 P47 BURLINSON LETTER BUGS BASEDIT AFFIX.CMD ON UNIBOARD SBC
APR 87 P47 WILLS LETTER FIX FOR TSC PR.CMD ADD CONTINUOUS COMMAND LINE BUFFER 6809 ASSEMBLY LANGUAGE
APR 87 P49 MOTOROLA ANNOUNCES M68HC05EVM 6805 EVALUATION KIT
MAY 87 P8 PASS C USER NOTES COMMAND LINE PROCESSING COMMA OPERATOR TOWER OF HANOI GAME C LISTING
MAY 87 P14 VOIGTS BASIC OS9 DOCUMENTATION STANDARDS LISTC LISTING
MAY 87 P19 ANDERSON USER NOTES WINDRUSH PLUS 68000 COMPILER PL9 LIBRARY BUG AUTOCAD CONE VOLUME BASIC LISTING
MAY 87 P22 LURIE FORTH TUTORIAL VIRTUAL MEMORY STRING CONSTANTS LITERALS BOOK REVIEW
MAY 87 P26 WELLER REVIEW SOFTOOLS FLEX UTILITES WRITTEN IN PL9 SOFTWARE TOOLS
MAY 87 P29 JONES ARTICLE TSC XBASIC EXPLAINED TOKEN HASHING TABLES ERROR MESSAGES STOP RND LIST
MAY 87 P45 KING ARTICLE SERIAL COMMUNICATION MOTOROLA SPI SCI 6805 MASTER SLAVE SCHEMATICS
MAY 87 P50 MACINTOSH SECTION STRIPPER CAUZIN SOFISTRIP SYSTEM
JUN 87 P8 PASS C USER NOTES C USERS GROUP ROUNDOFF ERRORS CONVERT ASSEMBLER EQUATE FILES TO C LISTING
JUN 87 P14 VOIGTS BASIC OS9 MACROS IN ASSEMBLERS EDITORS AND C
JUN 87 P19 ANDERSON USER NOTES WINDRUSH PLUS FOR 68000 PL9 PAT BUG STANDARDS OMEGASOFT PASCAL
JUN 87 P23 MACINTOSH SECTION 512K RAM MEMORY UPGRADE SCHEMATIC
JUN 87 P25 LURIE FORTH TUTORIAL STACKS
JUN 87 P28 STAFF REVIEW URDA P68000 MICROLAB NOTEBOOK COMPUTER

JUN 87 P38 PASS REVIEW ED 68000 TEXT EDITOR
JUN 87 P40 JONES ARTICLE TSC XBASIC EXPLAINED XPC PRECOMPILER XBASIC BUG CHAIN INT ASC VAL CHRS STRS
JUN 87 P49 MOTOROLA ANNOUNCES THE MC68606 MULTI LINK LAPD CONTROLLER
JUN 87 P51 DREXLER LETTER BUG FIX TIME.CMD (APR 87) REAL TIME CLOCK 6809 ASSEMBLY LISTING
JUL 87 P8 PASS C USER NOTES CONVERT TSC XBASIC TO C XPC COMPARE TWO FILES C LISTING
JUL 87 P13 VOIGTS BASIC OS9 DIRECTORIES INPUT BUFFER PROMPTED COPY PCOPY BASIC09 LISTING
JUL 87 P18 MACINTOSH SECTION MACLIGHTNING SPELLING CHECKER DICTIONARY THESAURUS
JUL 87 P21 LURIE FORTH TUTORIAL 6820 6821 PIA PARALLEL INTERFACE ADAPTER SWTPC MPLA IO FORTH LISTING
JUL 87 P25 WILLIAMS RAMBLINGS BOOK REVIEWS 68000 SYTEMS BY WILCOX MICROPROCESSOR SYSTEMS BY WIST & MEIKSIN COCO III
JUL 87 P28 STAFF REVIEW BARTON LABS LAB6809 SS50 CPU PROTOTYPING PICOBUG MONITOR
JUL 87 P38 REIMILLER PASCAL TUTORIAL CERTIFIED SOFTWARE CONVERT MICROWARE TO OMEGASOFT ASSEMBLER FILTER PASCAL LISTING
JUL 87 P41 LAVOREL PROGRAM PRICOL.BAS COLUMN PRINTING TSC XBASIC LISTING EPSON MX80
JUL 87 P43 JONES XBASIC EXPLAINED NUMBER BASE CONVERSIONS INPUT INPUTLINE LSET RSET OVERFLOW BUG FIX
AUG 87 P8 PASS C USER NOTES CONVERT BASIC TO C TSC XBASIC MICROWARE BASIC09 PAGINATE LIST OF FILES C LISTING
AUG 87 P14 VOIGTS BASIC OS9 SHELL CUSTOMIZING INPUT LINES BUILT IN COMMANDS DOUBLE STRIKE C LISTING
AUG 87 P19 MACINTOSH SECTION PAGE LAYOUT WORD PROCESSING READY SET GO
AUG 87 P21 LURIE FORTH TUTORIAL UTILITIES QX QUICK INDEX QLCOMPACT LIST FORTH LISTINGS
AUG 87 P25 REIMILLER PASCAL TUTORIAL OMEGASOFT EDITOR COMMANDS LINKAGE CREATOR COMPILER DEBUGGER
AUG 87 P28 JONES TUTORIAL ON DESIGN OF DIGITAL CONTROL CIRCUITS BOOLEAN LOGIC SYMBOLS
AUG 87 P42 JONES XBASIC EXPLAINED AND OR NOT SPLITTING OFF INSTRUCTIONS
AUG 87 P46 GREGORIE ARTICLE PL9 INTERFACE FOR ISAM DOCUMENTATION
AUG 87 P52 WILSON LETTER SHELL SORT 68000 ASSEMBLY LISTING
SEP 87 P8 PASS C USER NOTES CONVIERT BASIC TO C FILES OS9 MICROWARE C JUST BUG PL9 LISTING
SEP 87 P20 LURIE FORTH TUTORIAL CLEAR DISK COMMAND LINE ARGUMENTS PARSING FORTH LISTING
SEP 87 P22 JONES ARTICLE MATHEMATICAL DESIGN OF DIGITAL CONTROL CIRCUITS BOOLEAN ALGEBRA
SEP 87 P38 EPROM EMULATION FOR THE SS50 BUSS SCHEMATICS 6809 ASSEMBLY LISTING
SEP 87 P42 LAW MACINTOSH SECTION DOUG CLAPP'S WORD TOOLS COUNT SORT
SEP 87 P44 JONES XBASIC EXPLAINED SPEED AND SHORTEN PROGRAMS
OCT 87 P7 VOIGTS BASIC OS9 FILE ERRORS CRC VERIFY DATE C LISTING GETOPT
OCT 87 P12 PASS C USER NOTES CONVERTING BASIC TO C TABLE OF EQUIVALENTS STATEMENTS ECHO C LISTING
OCT 87 P18 ANDERSON USER NOTES PLUS PATOS68K FILE READ WRITE PORT PLUS TO SKDOS
OCT 87 P21 GROVES ARTICLE PASSWORD CHANGE OS9 BASIC09 LISTING
OCT 87 P26 WILLIAMS RAMBLINGS VAPORFACTS DESKTOP PUBLISHING
OCT 87 P28 ANCHOR MACINTOSH SECTION DARK CASTLE GAME
OCT 87 P38 KILLORAN ARTICLE TEXT HACKING COGNITIVE ENGINE CORP MLI MACROPROCESSOR
OCT 87 P40 JONES ARTICLE MATHEMATICAL DESIGN OF DIGITAL CONTROL CIRCUITS BINARY NUMBERS KARNAUGH MAP VETICH DIAGRAMS
OCT 87 P47 REIMILLER PASCAL TUTORIAL OMEGASOFT DEBUGGER
OCT 87 P50 ANNOUNCEMENT GESCOMP GESPAC G64 MICROCOMPUTER SYSTEM
OCT 87 P52 JONES XBASIC EXPLAINED MAX LINE LENGTH RANDOM FILES RBASIC
NOV 87 P7 PASS C USER NOTES REFERENCES BOOK REVIEWS EXTRACT STRINGS C LISTING
NOV 87 P12 VOIGTS BASIC OS9 MODULE DIRECTORY SYSCALL MDIR LINK GETMODDIR BASIC09 LISTING
NOV 87 P16 WEST ARTICLE 68XXX BOARDS ON THE STD BUSS
NOV 87 P19 JONES ARTICLE DIGITAL CONTROL DIRCUITS IMPLICANT CONTRADICTION AMBIGUITY COMBINATIONAL
NOV 87 P25 ALLAN ARTICLE CONVERT IBM XT KEYBOARD TO REPLACE CT82 SCHEMATIC 6805 ASSEMBLY LISTING
NOV 87 P40 LAW MACINTOSH SECTION CLIP ART WET ART WET PAINT HYPERCARD MULTIFINDER
NOV 87 P42 REIMILLER PASCAL TUTORIAL EXTERNAL PROCEDURE OR FUNCTION CALLS VALIDATE PASCAL LISTING
NOV 87 P45 LURIE FORTH TUTORIAL FIG FORTH INSTALLATION FLOORED MATH CASE EXECUTION TIMES
NOV 87 P48 CONDON ARTICLE BUILD THE GT4 GRAPHICS TERMINAL SCHEMATIC
NOV 87 P53 GREGORIE PL9 INTERFACE FOR THE ISAM CONT. LISTING
DEC 87 P7 PASS C USER NOTES PUBLIC DOMAIN PORTABLE MATH LIBRARY BY FRED FISH C LISTING MAC2UNIXC
DEC 87 P12 VOIGTS BASIC OS9 PASCAL P CODE SIMULTANEOUS EQUATION SOLUTION BY CRAMER'S RULE PASCAL LISTING
DEC 87 P17 JONES ARTICLE DIGITAL CONTROL CIRCUITS SEQUENTIAL FLOW TABLE RELAY OSCILLATOR
DEC 87 P23 MACINTOSH SECTION VIP TRANSLATORS FOR LIGHTSPEED C OR PASCAL LISTING
DEC 87 P37 WEST ARTICLE 68XXX AND THE STD BUSS CPU CARDS 2681 DUART SS30
DEC 87 P40 LURIE FORTH TUTORIAL MODULAR PROGRAMMING WORD NAMES FILE COPY UTILITY FORTH LISTING
DEC 87 P44 CONDON ARTICLE BUILD GT4 GRAPHIC TERMINAL SOFTWARE CONTROL FUNCTIONS
DEC 87 P48 BABIN ARTICLE LCD DRIVER WITH SERIAL INTERFACE CONTRAST VS VOLTAGE
DEC 87 P52 BILL WEST ANNOUNCES STD BUSS PRODUCTS STD020 STDZS STD08R 68020 68008
DEC 87 P53 MOTOROLA ANNOUNCES MC68HC05L6 MC68HC11 MC68606 MC68030 VME MVME374

+++



Dear Don,

So ... I was wondering if there's a Santa Claus out there somewhere, who'd maybe be interested in loaning me (or better still, donating) a Macintosh, with artwork or desk-top publishing capabilities. That would make it easier for me to create neat in-line diagrams instead of hand-drawn ones, do wrap-around text, etc, and also ease your editing burdens. Then I could plough ahead and complete the whole series with graphics built-in, and make it available via South East Media as a tutorial disk, or disks. We could also go ahead and maybe get it our as a book, in advance of the series itself getting completed!
In the meantime, for the benefit of all readers,

here's a summary of significant typos up to Mile 3, excluding obvious misspelling ones. which are not important anyway. Perhaps you could print this list as a supplement to one of my articles.

**Mile 00**
Diagram 1 Vertical dotted line should join the moveable parts of the relay-contacts.
Page 41 Expression at top should read Z1 = y1 + y2
TEST ONE Problem 2(ii) Coil should be labelled Y1
**Mile 01**
Page 23 Series Circuit should read 0.a = 0
Page 28 Line 6 should read [a'(b' + c')d']'.e
7 should read [a'(b' + c')d']
**Mile 02**
Solutions 2(a) should be Y = ab + bc + abc not Y1
there are two 3(e). Final one = 3(f)
below Diagram 6(d) should read -0— not -1—
page 45 para 3 should read One such is that as the ...
next para "block of four "bc'" not "bc"
**Mile 03**
Solutions (viii) should be a'bc' + bc'd + a'd + b'c
(don't know how I came to leave off all those primes!)

Sorry about all those errors. I guess if I didn't know the stuff in advance and therefore studied every dot and comma, these things would be less likely to get by me, but when I've spent hours composing each "Mile" my eyes are probably "screen-tired". Anyway, that's my excuse! I'll be in touch soon ... maybe with an additional chapter for "XBASIC XPLAINED". I have one in mind!
In the meantime, best wishes for a merry Xmas to all at 68MJ, and continued success for 1988.

Don Williams,
68 Micro Journal.
5900 Cassandra Smith Road,
Hixson, TN 37343

Sincerely,

Bob

R. Jones
President

**CERTIFIED SOFTWARE CORPORATION**
**616 Camino Caballo, Nipomo, CA 93444 USA**
**TEL:805-929-1395 TLX: 467013**

**FOR IMMEDIATE RELEASE**

Certified Software announces the introduction of it's OmegaSoft 68000 Industrial Pascal package specifically designed for the Atari ST computer with at least 1MB of memory. This version is customized for the Atari and includes a command line interpreter for greater flexibility and ease of use compared to other programs for the Atari that are hindered by the "desktop".

All Gemdos, Bios, Xbios, Line-A graphics, and AES/VDI calls are available as procedures and functions. In addition, the Pascal Shell has been modified so that the mouse can be used for menu selection. At least a double sided floppy drive is required, a hard disk is recommended but is not required for moderate sized programs.

The base package (PCSK-B) is priced at $290 and includes the compiler, relocatable macro assembler, linking loader, host debugger, pascal shell, screen editor, command line interpreter (with source code), and file patch utility. Options available are:

PCSK-R : source code for the runtime library. $100
PCSK-S : source code for the debugger, editor, pascal shell, and patch utility. $275
PCSK-T: target debugger with source code. $225

In West Germany, contact Byte Studio Borken (TEL: 02861-2147), all others contact Certified Software.

Mr. Don Williams
Editor
68 Micro Journal
5900 Cassandra Smith Road
Hixson, TN 37343

Dear Mr. Williams,

As of last year, I am still waiting to see full coverage for 68000 based systems. Last year I said that it would be my last unless you expanded to include more systems like Amiga and Atari. Well, you did make a plea for Atari help so I will stick it out for one more year. I own an Amiga so I expect progress on that system this year. Your management staff is not in step enough with the product world or users. otherwise they would have advertised for Atari much sooner. I expect to see the request for Amiga support soon. I have been a subscriber since 1976 and would not throw away a single issue. I have always enjoyed your publication until the last two years. I still have two old 6800 and one 6809 system, so the old issues are important to me. Please make the new issues important to my Amiga.

Respectfully,
Robert Hill

***Editor's Note:*** *Thanks Robert for letting us know how you felt. That is especially important to us. We have wanted to support the Atari and Amiga for sometime. However, as you probably know, there wasn't very much, if any, serious software for them. Today things are looking up. For instance, we are pretty excited about OS-9 running on the Atari, and even better, you choice of Personal or Professional versions. I call that serious software!*

*Now-here is the grabber. Without adequate input, articles, hints & kinks, tutorial and hands on material, and a lot more, it won't fly! We need your input and the input of folks like yourself who are interested in these systems, if it is to work.* ***We need volunteers willing to share their knowledge and experiences with all of us!*** *That's the way we have all been doing it for 10 years now.*

*Also you might note that our "Atari Call", elsewhere, has been changed to "Atari & Amiga".*

DMW

microware

MICROWARE SYSTEMS CORPORATION
1900 N.W. 114th Street
Des Moines, Iowa 50322

Phone 515-224-1929
Telex 910-520-2535
FAX 515-224-1282

FOR MORE INFORMATION CONTACT:
Mr. Andy Ball
Vice President, Marketing
Microware Systems Corporation
1900 NW 114th Street
Des Moines, Iowa 50322
515-224-1929

**MICROWARE ANNOUNCES ETHERNET SUPPORT FOR THE OS-9/68000 OPERATING SYSTEM**

Des Moines, Iowa. Microware Systems Corporation announces the availability of an Ethernet support package for the OS-9/68000 Operating System. The new Ethernet package utilizes the popular TCP/IP protocol to facilitate communications between OS-9/68000-based systems as well as other operating system environments.

The initial release of Microware's Ethernet protocol package is designed to support the Communication Machinery Corporation (CMC) ENP-10 Ethernet controller. Subsequent releases to support additional hardware, including the Motorola MVME 330-A-K controller, are u der development.

Microware's TCP/IP implementation conforms to the United States Department of Defense ArpaNet standards. This standard incorporates FTP (a popular file transfer protocol) and Telnet (a virtual terminal interface). FTP allows the user to transfer files and data between Ethernet nodes. elnet allows the user to login on remote systems over the Ethernet-TCP/IP network.

Microware selected the Ethernet protocol because of its rapid growth in popularity as a networking standard. he Microware Ethernet package allows the user to gateway from the OS-9 Operating System to UNIX, VMS and other operating system environments. This communications path provides connectivity for distributed program development and application systems.

The Microware Ethernet support package includes a new OS-9 file manager that supports a BSD 4.2 type socket interface and a new CMC ENP-10 device driver. Also included is complete documentation and 90-days free software support. Contact Microware, or an authorized distributor for pricing information.

**MICROWARE ANNOUNCES THE RELEASE OF MEMORY PROTECTION FOR THE OS-9/68000 OPERATING SYSTEM**

Des Moines, Iowa. Microware Systems Corporation announces the release of a memory protection module--System Protection Unit (SPU)--for the OS-9/68000 Operating System. SPU provides software memory protection for a Memory Management Unit and other custom devices.

The SPU module uses a system-wide permission mechanism to limit RAM access for user state tasks. Hardware devices supported by Microware's SPU module include the MC68451, MC68851 and custom devices. The SPU software is delivered as a modular addition to OS-9 in source code form and is divided by functions into separate routines that can be modified to support different memory protection devices.

Microware has designed SPU software to be transparent to the user. Only in cases of illegal memory access attempts--when bus errors occur--will the user become aware that SPU software is resident.

SPU memory protection provides both a secure environment for development system work and a significant aid in program debugging. The SPU module provides program run time protection, protection against wild pointers and detection problems before CPU crash.

The SPU software comes complete with a new permission memory map utility--MAPS. This utility graphically displays the memory address space using the OS-9 Termcap facility. MAPS displays information about process registers and the areas of RAM that each process can read and write.

The SPU module is immediately available under OEM license from Microware and authorized Microware distributors. Pricing is quantity dependent; contact Microware for more information.

The OS-9 Operating System is a real-tim , multi-user and multi-tasking system for computers based on the Motorola family of 68xxx processors. OS-9 is compact, ROMable and provides a UNIX-style environment for application software. Since its introduction in 1981, OS-9/68000 has been licensed to over 350 manufacturers world-wide for use in a variety of industrial, scientific and consumer products.

Founded in 1977, Microware System Corporation specializes in the development of advanced operating systems and programming languages. Last year Sony and Philips announced the OS-9 Operating System as the basis for Compact Disc-Interactive (CD-I) New Media technology.

**MICROWARE ANNOUNCES NEW C SOURCE-LEVEL DEBUGGER**

Des Moines, Iowa. Microware Systems Corporation announces the release of a new C source-level debugger for the OS-9/68000 Operating System. This new debugger represents a high-level language tool intended to decrease software development time and simplify programming. The C debugger contains features commonly found in C source-level debuggers, as well as a number of unique and innovative extensions designed for the OS-9 programming environment.

A powerful debugging tool in the C source-level debugger is a full-featured C expression interpreter. The programmer is able to call a function with parameters in the program being debugged. The result can then be printed, a break point set and the programmer can then step through the function. This allows the testing of functions separately in the program.

The expression interpreter also supports the new Microware C Compiler data types and operations (enumerators, bit fields, structure assignment, functions that return structures and structure parameters). This provides for total compatibility between the source level debugger and the C Compiler.

Another unique feature of the debugger is sub-line expression stepping. The debugger prints a character pointing to the expression in the source line that is to be executed next. If there is more than one statement in a line, then the debugger will step across the line to the next statement. This allows the programmer to watch each expression as it is executed in a complex C statement.

Microware has designed the debugger for optimum operator use. Each command contains both a short and long form, and multiple commands can be separated by a ";" on the same line. This relieves the programmer from the redundancy of keying in each command time after time. In addition, frequently used commands can be repeated simply by using the carriage return key.

Continued on page 56

# STYLOGRAPH

## WORD PROCESSING SYSTEM

### OS-9 LEVEL II VERSION

A Review
By: Bert Schneider

If you are not using Stylograph for your word processing and text editing, then you might as well pack up your computer, its operating system, and all of those neat utilities you have collected over the years and go back to playing games. Those are some serious words, but I mean business when it comes to the Stylograph Word Processing System.

In this review, I hope to give those of you out there not blessed with the opportunity of using any of Stylo Software's fantastic array of professional software, an overview of Stylograph, Mail Merge, and their Spelling Checker programs. I must point out that this package is for the Radio Shack Color Computer 3 and runs under Microware's OS-9 Level Two operating system. Its form, fit, and function is identical to the familiar Level One version for the Color Computer 1 and 2 as well as the standard OS-9 Stylograph. The major difference is you now can handle more text, the processor is twice as fast as the old Color Computer, you have 80 columns built into the system, and remote terminals are supported.

Having been very familiar with Level One Stylo running on my Color Computer 1 (using Wordpak from PBJ, INC.) I was anxious to have at least the same capability on my new Color Computer 3. I started working on my new system this past summer. You see my old system consisted of a 10 meg hard drive, one 40 track floppy, and all of the PBJ hardware that enabled me to do considerably more than just play games. This being America, and the term "new" being associated with better (so we are told from Madison Ave) I purchased a "new" Color Computer 3, an RGB monitor, and sold all of my old stuff. I am still in the process of building the expansion hardware to include two hardware serial ports, a parallel port, a real time clock, and a hard drive interface. The machine now resides in a PC case and is much more appealing than my old walnut case that was more comparable in size to the ENIAC!

Anyway, I tried to get some of the patches to allow me to use the old Stylo on my new found Level Two system. Fat chance! Although I heard of people doing it, I did not have any luck. Oh it did work, but the screen display was very interesting, to say the least. After I had used every profane word in the dictionary I called Roger at Stylo Software and found out about the new version.

#### OVERVIEW

Stylograph provides you with one of the most powerful word processing systems I have seen for a home computer system and outdoes several higher priced competitors. First and foremost, STYLO is a "What you see is what you get" full screen text editor. After every keystroke the screen is updated to reflect exactly the way the document will look like when it is printed. This means you do not have to use a text formatter after you have finished editing your work.

The screens are not user hostile. Editing is cursor based. A status line is provided to keep you informed of column, line, page, and mode information. Help screens are available at any time. And most commands other than format commands are entered from menu driven screens. Formatting is done with comma commands and control codes. The instant you enter a format command, the text display is updated to reflect the change. For example,

if you want to center a line, just enter the following

.ce
THIS IS CENTERING

and immediately the text is centered.

But editing is only one portion of STYLO. Since Stylograph is a word processing "system", you have more at your disposal than just text editing. From the main supervisory menu you may do any of the following: edit text, print out your text, save the text and return to OS-9, save your text to a predefined mark, return to OS-9, load a text file, erase current memory buffer, pass a command to OS-9, spool a file to disk for later printing, load a proportional spacing table for daisy wheel printers, or load in or edit a text file larger than the buffer. The spool function allows you to print a file in the background. Although I only recommend this if you use a ram disk or hard drive since it is very disk I/O intensive.

**MAIL MERGE**

Mail merge is useful for two reasons. The first and obvious reason for Mail Merge is to print out the same letter or form to many different destinations with different names and addresses. The other reason for using Mail Merge is to print out very large text files, larger than the buffer can handle at one time. You can even use Mail Merge in the background.

**SPELLING CHECKER**

The Stylo Spelling Checker compliments Stylograph and Mail Merge very well. If you have ever used a Spelling Checker this one is very straight forward. There are no fancy commands to remember. Just type the following and away you go:

OS9: spell /d1/letter

After that the program takes over and compares your text file "letter" to 42,000 words and is totally self-prompting.

The main dictionary can be manipulated to add or subtract words, and you can even develop your own personal supplemental dictionary for pronouns and specific terminology to fit your needs.

SPELL then displays to the standard output a word count, the number of different words used, and the number of misspelled words. All misspelled words are listed out for convenient reference.

You are next lead through a series of questions as to what you want done with the misspelled words. You may either ignore them, add them to the dictionary, flag them so you can search it out later, display the word in use to help you determine whether or not is was a misspelled word, change the word, or exit.

There are some optional utility commands that allow you to compress and decompress the dictionary and to add and subtract words from it.

An important note for all of these programs is that the utility STYFIX is used to configure each one of these programs onto your system disk.

STYLO for Level Two supports a remote terminal tied to a serial port and supports windows as well. This is a big departure from the Level One Color Computer System. First, the Level One system did not support a terminal. And naturally did not support windows. This new version does! I could not see using a terminal without the capability to edit text from a terminal. You could use build or edit, but somehow I could not see myself writing this review using edit or build! Second, Level One did not come with windows. This very important feature of the operating system is supported to give you a powerful software tool. Try this on some other machines. Compile a program in one window. Print out the error codes and lines in another window and then correct your errors in still another window while someone else is writing a novel on another terminal! Macintosh, eat your heart out!

Some of the terminal drivers included (a total of 34 are provided) are the ADM-3A, Heath/Zenith H19, Gimix OS9/window, Hazeltine 1500, DEC VT-52, H.P. 2621, Beehive 8100, and many more. You could modify any of the drivers, or add your own using STYFIX, the configuration program.

One neat feature built into this new program is a Math Package. That's right! Now you can add up your salary requirements while you edit your resume

all on the same machine. Just by typing "C" in the escape mode you have at your beck and call a calculator with addition, subtraction, multiplication, and division capabilities. You may also operate on rows or columns of numbers. An example might look something like this:

```
452
135
239
111
---
937
```

Just by entering control-Z and then pressing "C" it produces the total "937". You can only add or subtract in column mode however. After you have performed the operation you may go back and insert symbols or other text such as the following:

```
452
135
339
+ 11
---
937
```

**WHAT YOU GET**

Stylo comes with one floppy disk for each program (word processor, mail merge, and spelling checker each of which are sold separately or together) and a manual chock full of information.

Each disk comes with hard copy of the Read.me file located on each disk. The Read.me file describes the disk contents and how to get started along with some helpful notes. On each disk is the readme file, an installation program (to make it easier to install), a history file explaining corrections and modifications, a STY directory containing configuration files, help files, and a couple of sample letters. The CMDS directory obviously contains the program and STYFIX - the configuration program.

The manual includes an introduction and overview, a hands-on tutorial, specific mode and command explanations, and the OS-9 configuration. Several appendices are attached to help provide information on control and format commands, terminal and printer configurations, character modification codes, and information for changing text constants. A glossary is also included to help you understand wordprocessing and computer lingo.

Now since I have used Stylo before, I needed little introduction and instruction on the use of this program. I did however go through the entire manual along with its lessons. The manual is laid out very well and is straight forward. It provides the reader with just the right information at the right time but does give you more than enough information required to configure a terminal/printer, understand control codes and error messages, in a concise format in the appendices.

**CLOSING REMARKS**

On closing, I only want to say that I don't have any negative comments about STYLO SOFTWARE's products. Their software is top notch, their documentation is the finest, and their support can't be beat. I can say this. I can type well between 60 and 70 words a minute with few errors (touch typing) and Stylo keeps up with me without failure. There is only one product that I would recommend more than Stylograph and that is OS-9 itself!

So if you are a Color Computer 3 owner and either have OS-9 or are thinking about getting it, purchase Stylograph. You will save so much time and effort, you won't settle for anything less.

Stylograph, Mail Merge, and Spelling Checker are available from:

Avatlable from :

Southeast Media
5900 Cassandra Smith Rd.
Hixson, TN 37343
1-615-842-4600

Continued from page 52

The above options--in addition to the more fundamental features such as on-line help, tracing, break points, watch expressions and variable display/change facilities--make this new Microware C source-level debugger a powerful programming tool. The debugger not only speeds program development, but makes C language programming a tool that can be utilized by neophyte programmers.

The OS-9 Operating System is a real-time, multi-user and multi-tasking system for computers based on the Motorola family of 68xxx processors. OS-9 is compact, ROMable and provides a UNIX-style environment for application software. Since its introduction in 1983, OS-9/68000 has been licensed to over 350 manufacturers world-wide for use in a variety of industrial, scientific and consumer products.

# *ATARI & AMIGA CALL*

As most of you know, we are very sensitive to your wishes, as concerns the contents of these pages. One of the things that many of you have repeatedly written or called about is coverage for the **Atari & Amiga™** series of 68000 computers.

Actually we haven't been too keen on those systems due to a lack of serious software. They were mainly expensive "game-toy" systems. However, recently we are seeing more and more honest-to-goodness serious software for the Atari & Amiga machines. That makes a difference. I feel that we are ready to start some serious looking into a section for the Atari & Amiga computers. Especially so since **OS-9** is now running on the Atari (review copy on the way for evaluation and report to you) and rumored for the Amiga. Many of you are doing all kinds of interesting things on these systems. By sharing we all benefit.

**This I must stress - *Input from you on the Atari & Amiga.* As most of you are aware, we are a "contributor supported" magazine. That means that YOU have to do your part. Which is the way it has been for over 10 years. We need articles, technical, reviews of hardware and software, programming (all languages) and the many other facets of support that we have pursued for these many years. Also I will need several to volunteer to do regular columns on the Atari & Amiga systems. Without constant input we can't make it fly! So, if you do your part, we certainly will do ours. How about it, drop me a line or give me a phone call and I will get additional information right back to you. We need your input and support if this is to succeed!**

**DMW**



## Classifieds — As Submitted - No Guarantees

**MUSTANG-020** 16Mhz with 68881. OS9 Professional Package & C $3000. Call Tom (615)842-4600.

**AT&T 7300** UNIX PC, UNIX V OS, 1MB Memory, 20 MB Hard Disk, 5" Drive, Internal Modem, Mouse. Best Offer Gets It.

S+ System with Cabinet, 20 Meg Hard Disk & 8" Disk Drive with DMAF3 Controller Board. 1-X12 Terminal $4800.

**DAISY WHEEL PRINTERS**
Qume Sprint 9 - $900
Qume Sprint 5 - $800.

**HARD DISK 10 Megabyte Drive** - Seagate Model #412 $275.
3 - Dual 8" drive enclosure with power supply. New in box. $125 each.
5 - Siemens 8" Disk Drive. $100 each.
**Tano Outpost II**, 56K, 2 5" DSDD Drives. FLEX, MUMPS, $495.

**SWTPC S/09** with Motorola 128K RAM, 1-MPS2, 1-Parallell Port, MP-09CPU Card - $900 complete.
**Tom (615) 842-4600 M-F 9AM to 5PM EST**

•••

GMX MICRO - 20 (16.67 MHZ) with MC68881 and OS9/68020 professional pak $3500 or best offer.
John Bing 9-5 EST (301)428-8214

# SCULPTOR


**CUTS PROGRAMMING TIME UP TO 80%**

# 6809/68000-68030

# Save 70%

**Due to a "Special One Time" Purchase, We Are Making This Savings Offer. Quantities Limited!**

**See S. E. Media Catalog- page 29**

***Once this supply is gone - the price goes back up!***

**System OS-9: 6809/68000-68030**

**• Regular ~~$995.00~~**

**ONLY $295.00**

+ $7.50 S&H USA
Overseas - see S.E. Media Shipping Rates- page 29

**$AVE - WHILE SUPPLIES LAST!**

# SK★DOS

*The Generic DOS™* for 68000 applications in

- Industrial Control
- Business Use
- Educational Computing
- Scientific Computing
- Number Crunching
- Dedicated Systems
- Turnkey Systems
- Data Collection
- Single-board Computers
- Bus-oriented Computers
- Graphics Workstations
- One-of-a-kind Systems
- Advanced Hobbyist Use

SK•DOS is a single-user disk operating system for computers using Motorola 32 bit CPUs such as the 68008, 68000, 68010, and 68020. It provides the power of a full OS9, yet is simple and easy to use, and will run on systems from 32K to 16 megabytes. Because **SK•DOS** is easily implemented on a new system, we call it "The Generic DOS" which allows programs written for one system to be run on many others.

SK•DOS comes with over 40 commands and system programs, including a 6809 emulator which allows 68K SK•DOS to run application programs and languages developed for 6809 **SK•DOS** and other systems. Assemblers, editors, and higher level language support are available from third party software vendors and through public domain software.

SK•DOS is available for single copy or dealer sales, as well as OEM licensing. Single copies cost $125 (inquire as to available systems). Extremely attractive OEM licensing terms are also available. An optional Configuration Kit contains a detailed Configuration Manual and two disks of source code for system adaptation, including source code for a system monitor/debug ROM and other programs useful for adapting SK•DOS to new systems.

SK•DOS
*is available from*

STAR-K
SOFTWARE SYSTEMS CORPORATION
P.O. BOX 209 · MT. KISCO, NY 10549 · 914/241-0287
TELEX 5106016774

## INDUSTRIAL PASCAL FOR 68000 AND 6809

**PCSK** is a package that generates code for a 68000 series processor running on a 68000 development system. It includes the compiler, assembler, linker, host debugger, target debugger, and screen editor, all integrated together and controlled by a menu driven shell program. Source code is included for the runtime library and many of the utilities. Host operating systems supported are OS-9/68000 (Microware), PDOS (Eyring Research), and VERSAdos (Motorola).

**PXK9** is a package that generates code for a 6809 processor running on a 68000 development system. Includes all of the features of the PCSK package above, except for the host debugger. Host operating system is OS-9/68000.

### I WANT IT, WHERE DO I GET IT?

For more information on either of these two products please contact Certified Software, South East Media, or one of our European Licensees.

**OEM LICENSEES**

Gespac sa, 3, chemin des Aulx, CH-1228 Geneva/Plan-les-Ouates, Switz. TEL (022) 713400, TLX 429989

PEP Elektronik Systeme GmbH, Am Klosterwald 4, D-8950 Kaufbeuren, West Germany. TEL (08341) 8974, TLX 541233.

Ellec Elektronik GmbH, Galileo-Galilei-Strasse, 6500 Mainz 42, Postfach 65, West Germany. TEL (06131) 50031, TLX 4187273.

**DISTRIBUTORS**

R.C.S. Microsystems Ltd. 141 Uxbridge Road, Hampton Hill, Middlesex, England. TEL 01-9792204, TLX 8951470.

Dr. Rudolf Keil GmbH, Porphystrasse 15, D-6905 Schriesheim, West Germany. TEL 062 03/6741, TLX 465025.

Elsolt AG, Zelgweg 12, CH-5405 Baden-Daettwil, Switzerland. TEL 056-833377, TLX 828275.

Byte Studio Borken, Butenwall 14, D-4280 Borken, West Germany. TEL 02861-2147, TLX 813343.

**CERTIFIED SOFTWARE CORPORATION**
616 CAMINO CABALLO, NIPOMO, CA 93444
TEL: (805) 929-1395 TELEX: 467013
FAX: (805) 929-1395 (M10-8AM)

## SOFTWARE FOR 680x AND MSDOS

**SUPER SLEUTH DISASSEMBLERS**
**EACH $99-FLEX $101-OS9 $100-UNIFLEX**
**OBJECT-ONLY versions: EACH $50-FLEX,OS9,COCO**
interactively generate source on disk with labels, include xref, binary editing
specify 6800,1,2,3,5,8,9/6502 version or Z80/8080,5 version
OS9 version also processes FLEX format object file under OS9
COCO DOS available in 6800,1,2,3,5,8,9/6502 version (not Z80/8080,5) only
**68010 disassembler $100-FLEX,OS9,UNIFLEX,MSDOS,UNIX,SKDOS**

**CROSS-ASSEMBLERS WITH MACRO CAPABILITIES**
**EACH $30-FLEX,OS9,UNIFLEX,MSOOS,UNIX,SKOOS 3/$100 ALL/$200**
specify: 180c,6502,6801/11,6804,6805,6809,Z8,Z80,8048,8051,8085,68010,32000
modular cross-assemblers in C, with load/unload utilities
sources for additional $50 each, $100 for 3, $300 for all

**DEBUGGING SIMULATORS FOR POPULAR 8-BIT MICROPROCESSORS**
**EACH $75-FLEX $100-OS9 $80-UNIFLEX**
**OBJECT-ONLY versions: EACH $50-COCO FLEX,COCO OS9**
interactively simulate processors, include disassembly formatting, binary editing
specify for 6800/1, (14)6805, 6502, 6809 OS9, Z80 FLEX

**ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809**
6502 to 6809 $75-FLEX $85-OS9 $80-UNIFLEX
6800/1 to 6809 & 6809 to position-ind. $50-FLEX $75-OS9 $60-UNIFLEX

**FULL-SCREEN XBASIC PROGRAMS with cursor control**
**AVAILABLE FOR FLEX, UNIFLEX, AND MSOOS**

| | |
|---|---|
| DISPLAY GENERATOR/DOCUMENTOR | $50 w/source, $25 without |
| MAILING LIST SYSTEM | $100 w/source, $50 without |
| INVENTORY WITH MRP | $100 w/source, $50 without |
| TABULA RASA SPREADSHEET | $100 w/source, $30 without |

**DISK AND XBASIC UTILITY PROGRAM LIBRARY**
**$50-FLEX $30-UNIFLEX/MSDOS**
edit disk sectors, sort directory, maintain master catalog, do disk sorts, resequence some or all of BASIC program, xref BASIC program, etc.
non-FLEX versions include sort and resequencer only

**CMODEM TELECOMMUNICATIONS PROGRAM**
**$100-FLEX,OS9,UNIFLEX,MS-OOS,UNIX,SKDOS**
**OBJECT-ONLY versions: EACH $50**
menu-driven with terminal mode, file transfer, MODEM7, XON-XOFF, etc.
for COCO and non-COCO; drives internal COCO modem port up to 2400 Baud

## DISKETTES & SERVICES

**5.25" DISKETTES**
**EACH 10-PACK $7.50-SSSO/SSDO/OSOO**
American made, guaranteed 100% quality, with Tyvek jackets, hub rings, and labels

**ADDITIONAL SERVICES FOR THE COMPUTING COMMUNITY**

**CUSTOMIZED PROGRAMMING**
we will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the marketability of the modifications.

**CONTRACT PROGRAMMING**
we will create new programs or modify existing programs on a contract basis. a service we have provided for over twenty years; the computers on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 6502, 68k9, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task.

**CONSULTING**
we offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses.

**Computer Systems Consultants, Inc.**
**1454 Latta Lane, Conyers, GA 30207**
**Telephone 404-483-4570 or 1717**

**We take orders at any time, but plan long discussions after 8, if possible.**

**Contact us about catalog, dealer, discounts, and services. Most programs in source: give computer, OS, disk size. 25% off multiple purchases of same program on one order. VISA and MASTER CARD accepted; US funds only, please. Add GA sales tax (if in GA) and 5% shipping.**
(UNI)FLEX tm Technical Systems Consultants; OS/9 Microware; COCO Tandy, MSDOS Microsoft, SKDOS Star-K Software

# THE 6800-6809 BOOKS

## ..HEAR YE.....HEAR

## OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibbles OS9 USER NOTES

**Information for the BEGINNER to the PRO, Regular or CoCo OS9**

**Using OS9**
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS, Generating a New Bootstrap, Building a new System Disk, OS9 Users Group, etc.

**Program interfacing to OS9**
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

**Programming Languages**
Assembly Language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

**Disks Include**
**No typing all the Source Listings in.** Source Code and, where applicable, assembled or compiled **Operating Programs.** The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point" for developing your **OWN** more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

**BOOK $9.95**

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)
**Deluxe Binder** - - - - - - - - - - $5.50

**All Source Listings on Disk**

1-8" SS, SD Disk - - - - $14.95
2-5" SS, DD Disks - - - $24.95

## FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor profesionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks

| | |
|---|---|
| LOGO.C1 | File load program to offset memory — ASM PIC |
| MEMOVE.C1 | Memory move program — ASM PIC |
| DUMP.C1 | Printer dump program — uses LOGO — ASM PIC |
| SUBTEST.C1 | Simulation of 6800 code to 6809, show differences — ASM |
| TERMEM.C2 | Modem input to disk (or other port input to disk) — ASM |
| M.C2 | Output a file to modem (or another port) — ASM |
| PRINT.C3 | Parallel (enhanced) printer driver — ASM |
| MODEM.C2 | TTL output to CRT and modem (or other port) — ASM |
| SCIPKG.C1 | Scientific math routines — PASCAL |
| U.C4 | Mini-monitor, disk resident, many useful functions — ASM |
| PRINT.C4 | Parallel printer driver, without PFLAG — ASM |
| SET.C5 | Set printer modes — ASM |
| SETBAS1.C5 | Set printer modes — A-BASIC |

NOTE: .C1,.C2, etc.=Chapter 1, Chapter 2, etc.

**Over 30 TEXT files included in ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

**Shipping & Handling $3.50 per Book, $2.50 per Disk set**
**Foreign Orders Add $4.50 Surface Mail**
**or $7.00 Air Mail**
**If paying by check - Please allow 4-6 weeks delivery**
**• All Currency in U.S. Dollars**

## Continually Updated In 68 Micro Journal Monthly

**Computer Publishing Inc.**
**5900 Cassandra Smith Rd.**
**Hixson, TN 37343**

**(615) 842-4601**
**Telex 5106006630**

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware and Motorola
"68' Micro Journal is a trademark of Computer Publishing Inc.

# !!! Subscribe Now !!!
# 68 MICRO JOURNAL

**OK, PLEASE ENTER MY SUBSCRIPTION**

Bill My: Mastercard ☐ VISA ☐

Card #_____ Exp. Date_____

For 1 Year ____ 2 Years ____ 3 Years ____

Enclosed: $_____

Name _____

Street _____

City _____ State ____ Zip _____

Country _____

My Computer Is: _____

**Subscription Rates**

**U.S.A.: 1 Year $24.50, 2 Years $42.50, 3 Years $64.50**

***Foreign Surface: Add $12.00 per Year to USA Price.**

***Foreign Airmail: Add $48.00 per Year to USA Price.**

***Canada & Mexico: Add $9.50 per Year to USA Price.**

***U.S. Currency Cash or Check Drawn on a USA Bank !**

**68 Micro Journal**
5900 Cassandra Smith Rd.
POB 849
Hixson, TN 37343

**Telephone 615 842-4600**
**Telex 510 600-6630**

## Reader Service Disks

| | |
|---|---|
| Disk- 1 | Filesort, Minicat, Minicopy,Minifms, **Lifetime, **Poetry, **Foodlist, **Diet. |
| Disk- 2 | Diskedit w/ inst.& fixes, Prime, *Prmod, **Snoopy, **Football, **Hexpawn,**Lifetime. |
| Disk- 3 | Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave. |
| Disk- 4 | Mailing Program, *Finddat, *Change, *Testdisk. |
| Disk- 5 | *DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING. |
| Disk- 6 | **Purchase Order, Index (Disk file indx). |
| Disk- 7 | Linking Loader, Rload, Hextness. |
| Disk- 8 | Crtest, Cmpher (May 82). |
| Disk- 9 | Datacopy, Diskfix9 (Aug 82). |
| Disk-10 | Home Accounting (July 82). |
| Disk-11 | Dissembler (June 84). |
| Disk-12 | Modem68 (May 84). |
| Disk-13 | *Initmf68, Testmf68,*Cleanup,*Dskalign,Help,Date.Txt. |
| Disk-14 | *Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib. |
| Disk-15 | Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo). |
| Disk-16 | Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc. |
| Disk-17 | Match Utility, RATBAS, A Basic Preprocessor. |
| Disk-18 | Parse.Mod, Size.Cmd (Sept. 85 Armstrong),CMDCODE, CMD.Txt (Sept. 85 Spray). |
| Disk-19 | Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc. |
| Disk-20 | UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist). Dragon.C, Grep.C, LS.C, FDUMP.C. |
| Disk-21 | Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more. |
| Disk-22 | Read CPM & Non-FLEX Disks. Fraser May 1984. |
| Disk-23 | ISAM, Indexed Sequential file Accessing Methods, Condon Nov. 1985. Extensible Table Driven. Language Recognition Utility, Anderson March 1986. |
| Disk-24 | 68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current. |
| Disk-25 | KERMIT for FLEX derived from the UNIX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk. |
| Disk-26 | Compacta UniBoard review, code & diagram, Burlison March '86. |
| Disk-27 | ROTABIT.TXT, SUMSTEST.TXT, CONDATA.TXT, BADMEN.TXT. |
| Disk-28 | CT-82 Emulator, bit mapped. |
| Disk-29 | **Star Trek |
| Disk-30 | Simple Winchester, Dec.'86 Green. |
| Disk-31 | *** Read/Write MS/PC-DOS (SK*DOS) |
| Disk-32 | Heir-UNIX Type upgrade - 68MJ 2/87 |
| Disk-33 | Build the GT-4 Terminal - 68MJ 11/87 Condon. |

NOTE:
This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 - ** Denotes BASIC
*** Denotes 68000 - 6809 no indicator.

VISA

**8" disk $19.50**
**5" disk $16.95**

MasterCard

Shipping & Handling -U.S.A. Add: - $3.50
Overseas add: $4.50 Surface - $7.00 Airmail

## 68 MICRO JOURNAL
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4600 - Telex 510 600-6630

# GMX MICRO-20 and TWINGLE-20 PRICE LIST

## *All versions include 1 SAB Board*

| | MICRO-20 with 1MB RAM | MICRO-20 with 2MB RAM | TWINGLE-20 with 4MB RAM |
|---|---|---|---|
| 12.5 MHz | 1855.00 | 2155.00 | 3855.00 |
| 16.67 MHz | 2185.00 | 2485.00 | 4185.00 |
| 20 MHz | 2585.00 | 2885.00 | 4785.00 |

### OPTIONAL PARTS AND ACCESSORIES

68881 12.5MHz Floating Point Coprocessor ..... $ 165.00
68881 16.67MHz Floating Point Coprocessor ..... $ 225.00
68881 20MHz Floating Point Coprocessor ..... $ 345.00
MOTOROLA 68020 USERS MANUAL ..... $ 18.00
MOTOROLA 68030 USERS MANUAL ..... $ 18.00
MOTOROLA 68881 USERS MANUAL ..... $ 18.00

**SBC ACCESSORY PACKAGE (M20-AP)** ..... $1399.00

The package includes a PC-style cabinet with a custom backpanel, a 25 Megabyte (unformatted) hard disk and controller, a floppy disk drive, a 150 watt power supply, cooling fan, panel mounted reset and abort switches, and all necessary internal cabling. (For use with SAB—90 serial connectors only.)

2nd 5"80 FLOPPY & CABLES FOR M20-AP, ADD ..... $ 250.00
SECOND 25MB HARD DISK & CABLES, ADD ..... $ 780.00
TO SUBSTITUTE 50MB HD FOR 25MB HD, ADD ..... $ 290.00
TO SUBSTITUTE 80MB HD FOR 25MB HD, ADD ..... $1500.00
TO SUBSTITUTE 155MB FOR 25MB HD, ADD ..... $2100.00
60MB TEAC STREAMER WITH ONE TAPE ..... $ 690.00
PKG. OF 5 TEAC TAPES ..... $ 112.50

CUSTOM BACK PANEL PLATE (BPP-PC) ..... $ 44.00

### I/O EXPANSION BOARDS

**16 PORT SERIAL BOARD ONLY (SBC-16S)** ..... $ 335.00

The SBC-16S extends the I/O capabilities of the GMX Micro-20 68020 Single-board Computer by adding sixteen asynchronous serial I/O ports. By using two SBC-16S boards, a total of thirty-six serial ports are possible.

**RS232 ADAPTER (SAB-25, SAB-9D or SAB-8M)** ..... $165.00

The board provides level-shifting between TTL level and standard RS-232 signal levels for up to 4 serial I/O ports.

**60 LINE PARALLEL I/O BOARD (SBC-60P)** ..... $398.00

The GMX SBC-60P uses three 68230 Parallel Interface/Timers (PI/Ts) to provide up to forty-eight parallel I/O lines. The I/O lines are buffered in six groups of eight lines each, with separate buffer direction control for each group. Buffer direction can be fixed by hardware jumpers, or can be software programmable for bidirectional applications.

**PROTOTYPING BOARD (SBC-WW)** ..... $75.00

The SBC-WW provides a means of developing and testing custom I/O interface designs for the GMX Micro-20 68020 Single-board Computer. The board provides areas for both DIP (Dual Inline Package) and PGA (Pin Grid Array) devices, and a pre-wired memory area for up to 512K bytes of dynamic RAM.

**I/O BUS ADAPTER (SBC-BA)** ..... $195.00

The SBC-BA provides an interface between the GMX Micro-20 68020 Single-board Computer and the Motorola Input/Output Channel (I/O bus). With the I/O bus, up to sixteen off-the-shelf or custom peripheral devices (I/O modules) can be connected to the GMX Micro-20.

**ARCNET LAN board w/o Software (SBC-AN)** ..... $475.00

The SBC-AN provides an interface between the GMX Micro-20 68020 Single-board Computer and the ARCNET modified token-passing Local Area Network (LAN) originally developed by Datapoint Corp. The ARCNET is a baseband network with a data transmission rate of 2.5 Megabits/second. The standard transmission media is a single 93 ohm RG-62/U coaxial cable. Fiber optic versions are available as an option.

**OS9 LAN Software Drivers for SBC-AN** ..... 120.00

### GMX MICRO-20 SOFTWARE

**020 BUG UPDATE — PROMS & MANUAL** ..... $150.00

***THESE 68020 OPERATING SYSTEMS ARE PRICED WHEN PURCHASED WITH THE MICRO-20. PLEASE ADD $150.00 IF PURCHASED LATER FOR THE UPDATED PROMS AND MANUALS. ALL SHIPPED STANDARD ON 5¼" DISKS 3½" OPTIONAL IF SPECIFIED.***

**OS9/68020 PROFESSIONAL PAK** ..... $850.00

Includes O.S., "C", uMACS EDITOR, ASSEMBLER, DEBUGGER, development utilities, 68881 support.

**OS9/68020 PERSONAL PAK** ..... $ 400.00

Personal OS-9 systems require a GMX Micro-20 development system running Professional OS-9/68020 for initial configuration.

**BASIC** (included in PERSONAL PAK) ..... $ 200.00
**C COMPILER** (included in PROFESSIONAL PAK) ..... $ 750.00
**PASCAL COMPILER** ..... $ 500.00

UniFLEX (for Micro-20) ..... $ 400.00
**UniFLEX WITH REAL-TIME ENHANCEMENTS** ..... $ 800.00
**UniFLEX VM** (for TWINGLE-20) ..... $ 600.00
**UniFLEX VM REAL-TIME ENHANCEMENTS** ..... $1000.00

#### Other Software for UniFLEX

**UniFLEX BASIC W/PRECOMPILER** ..... $ 300.00
**UniFLEX C COMPILER** ..... $ 350.00
**UniFLEX COBOL COMPILER** ..... $ 750.00
**UniFLEX SCREEN EDITOR** ..... $ 150.00
**UniFLEX TEXT PROCESSOR** ..... $ 200.00
**UniFLEX SORT/MERGE PACKAGE** ..... $ 200.00
**UniFLEX VSAM MODULE** ..... $ 100.00
**UniFLEX UTILITIES PACKAGE I** ..... $ 200.00
**UniFLEX PARTIAL SOURCE LICENSE** ..... $1000.00

***GMX EXCLUSIVE VERSIONS, CUSTOMIZED FOR THE MICRO-20, OF THE BELOW LANGUAGES AND SOFTWARE ARE ALSO AVAILABLE FROM GMX.***

**ABSOFT FORTRAN (UniFLEX)** ..... $1500.00
**SCULPTOR (specify UniFLEX or OS9)** ..... $ 995.00
**FORTH (OS9)** ..... $ 595.00
**DYNACALC (specify UniFLEX or OS9)** ..... $ 300.00

**GMX DOES NOT GUARANTEE PERFORMANCE OF ANY GMX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.**

**ALL PRICES ARE F.O.B. CHICAGO IN U.S. FUNDS**

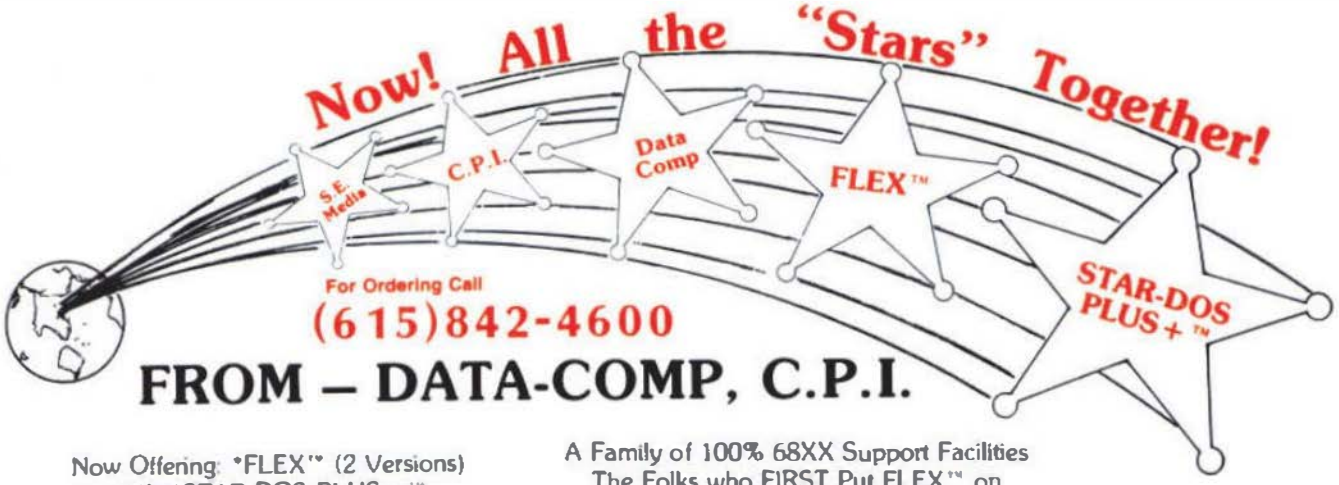
GMX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account number 73-32033.

CONTACT GMX FOR MORE INFORMATION ON THE ABOVE PRODUCTS

GMX STILL SELLS GIMIX S50 BUS SYSTEMS, BOARDS & PARTS. CONTACT GMX FOR COMPLETE PRICE LIST.

**GMX** 1337 W. 37th Place, Chicago, IL 60609 (312) 927-5510 — TWX 910-221-4055 — FAX (312) 927-7352




# FROM – DATA-COMP, C.P.I.

Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

**FLEX-CoCo Sr.**
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00 **Only $79.95**

**STAR-DOS PLUS+**
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

**$34.95**

**FLEX-CoCo Jr.**
without TSC Editor & Assembler
**$49.95**

**PLUS**

**ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE**

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

**TSC Editor**
Reg $50.00
**NOW $35.00**

**TSC Assembler**
Reg $50.00
**NOW $35.00**

## CoCo Disk Drive Systems

| | |
|---|---|
| 2 THINLINE DOUBLE SIDED DOUBLE DENSITY DISK DRIVES SYSTEM WITH POWER SUPPLY,CABINET,DISK DRIVE CABLE,J&M NEW DISK CONTROLLER JFD-CP WITH J-DOS,RS-DOS OPERATING SYSTEMS. | $469.95 |
| * Specify What CONTROLLER You Want J&M, or RADIO SHACK | |
| THINLINE DOUBLE SIDED DOUBLE DENSITY 40 TRACKS | $129.95 |

## Verbatim Diskettes

| | |
|---|---|
| Single Sided Double Density | $ 24.00 |
| Double Sided Double Density | $ 24.00 |

## Controllers

| | |
|---|---|
| J&M JFD-CP WITH J-DOS | $139.95 |
| WITH J-DOS, RS-DOS | $159.95 |
| RADIO SHACK 1.1 | $134.95 |
| RADIO SHACK Disk CONTROLLER 1.1 | $134.95 |

## Disk Drive Cables

| | |
|---|---|
| Cable for One Drive | $ 19.95 |
| Cable for Two Drives | $ 24.95 |

## MISC

| | |
|---|---|
| 64K UPGRADE FOR C,D,E,P, AND COCO 11 | $ 29.95 |
| RADIO SHACK BASIC 1.2 | $ 24.95 |
| RADIO SHACK DISK BASIC 1.1 | $ 24.95 |
| DISK DRIVE CABINET FOR A SINGLE DRIVE | $ 49.95 |
| DISK DRIVE CABINET FOR TWO THINLINE DRIVES | $ 69.95 |

**PRINTERS**

| | |
|---|---|
| EPSON LX-80 | $289.95 |
| EPSON MX-70 | $325.95 |
| EPSON MX-100 | $495.95 |

**ACCESSORIES FOR EPSON**

| | |
|---|---|
| 8148 2K SERIAL BOARD | $ 89.95 |
| 8149 32K EXPAND TO 128K | $169.95 |
| EPSON MX-RX-80 RIBBONS | $ 7.95 |
| EPSON LX-80 RIBBONS | $ 5.95 |
| TRACTOR UNITS FOR LX-80 | $ 39.95 |
| CABLES & OTHER INTERFACES CALL FOR PRICING | |

**DATA-COMP**
5900 Cassandra Smith Rd.
Hixson, TN 37343

MasterCard VISA

**SHIPPING**
USA ADD 2%
FOREIGN ADD 5%
MIN. $2.50

(615)842-4600
For Ordering
Telex 5106006630

# An Ace of a System in Spades! The New

# MUSTANG-08/A™

**Now with 4 serial ports standard & speed increase to 12 Mhz CPU + on board battery backup and includes the PROFESSIONAL OS-9 package - including the $500.00 OS-9 C compiler! This offer won't last forever!**

## *NOT* 128K, *NOT* 512K
## *FULL* 768K No Wait RAM

The MUSTANG-08™ system took every hand from all other 68008 systems we tested, running OS-9 68K!

The MUSTANG-08 includes OS9-68K™ and/or Peter Stark's SK*DOS™. SK*DOS is a single user, single tasking system that takes up where *FLEX™ left off. SK*DOS is actually a 68XXX FLEX type system *(Not a TSC product.)*

The OS-9 68K system is a full blown multi-user, multi-tasking 68XXX system. All the popular 68000 OS-9 software runs. It is a speed whiz on disk I/O. Fact is the MUSTANG-08 is faster on disk access than some other 68XXX systems are on memory cache access. Now, that is fast! And that is just a small part of the story! See benchmarks!

SALE!!

System includes OS-9 68K or SK*DOS - Your Choice

Specifications:

| | | |
|---|---|---|
| CPU | MC68008 | 12 Mhz |
| RAM | 768K | 256K Chips |
| | No Wait States | |
| PORTS | 4 - RS232 | MC68681 DUART |
| | 2 - 8 bit Parallel | MC6821 PIA |
| CLOCK | MK48T02 | Real Time Clock Bat. B/U |
| EPROM | 16K, 32K or 64K | Selectable |
| FLOPPY | WD1772 | 5 1/4 Drives |
| HARD DISK | Interface Port | WD1002 Board |

**Now more serial ports - faster CPU Battery B/U - and $850.00 OS-9 Professional with C compiler included!**

***$400.00** See Mustang-02 Ad - page 5 for trade-in details


**MUSTANG-08**

**LOOK**

| | | | | Seconds | 32 bit Integer | Register Long |
|---|---|---|---|---|---|---|
| Other | 68008 | 8 Mhz | OS-9 | 68K | 18.0 | 9.0 |
| MUSTANG-08 | 10 Mhz | | OS-9 | 68K | 9.8 | 6.3 |

```
Main()
{
C Benchmark Loop        /* int i; */
                        register long i;
                        for (i=0; i < 999999; ++i);
}
```

## Now even faster! with 12 Mhz CPU

| C Compile times: OS-9 68K Hard Disk | | |
|---|---|---|
| MUSTANG-08 | 8 Mhz CPU | 0 min - 32 sec |
| Other popular 68008 system | | 1 min - 05 sec |
| MUSTANG-020 | | 0 min - 21 sec |

## 25 Megabyte Hard Disk System

# $2,398.90

**Complete with PROFESSIONAL OS-9 includes the $500.00 C compiler, PC style cabinet, heavy duty power supply, 5" DDDS 80 track floppy, 25 MegByte Hard Disk - Ready to Run**

Unlike other 68008 systems there are several significant differences. The MUSTANG-08 is a full 12 Megahertz system. The RAM uses NO wait states, this means full bore MUSTANG type performance.

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 Megabytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system.

A RAM disk of 480K can be easily configured, leaving 288K free for program/system RAM space. The RAM DISK can be configured to any size your application requires (system must have 128K in addition to its other requirements). Leaving the remainder of the original 768K for program use. Sufficient source included (drivers, etc.)

FLEX is a trademark of TSC  MUSTANG-08 is a trademark of CPI

## Data-Comp Division

CPI — A Decade of Quality Service* — Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road

Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

* Those with SWTPC Hi-density FLEX 5" - Call for special info.